A novel design technique and hardware implementation of digital filters for high-speed Realtime filtering.

So, J. L-W

A NOVEL DESIGN TECHNIQUE AND HARDWARE IMPLEMENTATION OF DIGITAL FILTERS FOR HIGH-SPEED REALTIME FILTERING

BY

JOHN LING-WING SO

A thesis submitted to the University Of London for the degree of Doctor Of Philosophy.

Electronics Research
Chelsea College
University Of London
Pulton Place
London SW6

August 1978

ACKNOWLEDGEMENT

I would like to thank Mr. C.S. Aitchison for his guidance and helpful discussions throughout the duration of the project. His suggestions in the preparation of this thesis are also cordially acknowledged.

I would also like to extend my gratitude to some of the staff and research students at Pulton Place Electronics Research Laboratory for their discussions on various matters.

Thanks are also due to Dr. A.C. Davies, at The City University, London, for his occasional guidance.

Finally, I am greatly indebted to my dear father and my family for their endurance and encouragement throughout the duration of the project and the preparation of this thesis.

ABSTRACT

A design technique, which is particularly suitable for designing filter banks, is proposed for high-speed realtime digital filtering. In this design technique, the use of a proposed frequency-sampling procedure results in a closed-form expression for a linear-phase equiripple passband structure which is further optimized by a proposed optimization procedure to yield a final optimum transfer function. Furthermore, the proposed design technique offers a reduction in hardware, even when implemented with existing approaches in hardware implementation. This is feasible via the modularity of the resultant design which allows both zero- and pole-sharing as well as realtime computations of filter coefficients, so as to reduce the storage requirement of, for instance, a filter bank implementation.

In addition to the above proposed design technique, a hardware implementation of digital filters for high-speed realtime filtering has also been proposed. The proposed hardware implementation does not include "adds" and "multiplies" of conventional arithmetic units, but instead, employs an algorithm, which we have called the "Carry Brought Forward Compensation Scheme" (CBFCS), in order to implement a computational unit, which we have called the "Adderless-Multiplierless Unit" (AMU), to replace conventional arithmetic units, substituting table lookup for computation. Due to the rapidly increasing speed and capacity of semiconductor memory per unit of power dissipation, the proposed hardware implementation capitalizes on memory to offer significant reductions in cost and power consumption for the same speed of operation as that of existing approaches. In fact, the proposed "Carry Brought Forward Compensation Scheme" makes possible operational speeds which cannot be achieved by existing approaches by "sectioning" the carry propagation path which is the fundamental limitation in speed of all conventional processors. Furthermore, a generalized quantization noise model is subsequently proposed for the analysis of finite-word-length effects on the above proposed hardware implementation. Additionally, a finite-word-length demonstration processor was built to verify experimentally the proposed hardware implementation and to investigate the above generalized quantization noise model. The logic design of the above demonstration processor

is at system level and the resultant architecture is very suitable for large-scale-integration. Finally, results from simulations and experiments with the finite-word-length demonstration processor showed a satisfactory agreement between theory and practice as far as the finite accuracy of the processor allowed.

CONTENTS PAGE

CHAPTER ONE

INTRODUCTION

1.1 HISTORICAL BACKGROUND: (1),(2),(3)

In general, one can speak of a digital filter as a computational process or algorithm by which a digital signal or a sequence of quantized numbers, acting as an input signal x_n, is transformed by some well-defined rules into a second sequence of quantized numbers, termed the output digital signal y_n (9). The algorithm may be implemented in software as a computer subroutine for a general-purpose computer, or in hardware as a special-purpose processor. The term digital filter is then specifically applied to the particular routine in execution or to the dedicated digital hardware.

While seismologists made notable use of digital filter concepts to solve many interesting problems, it was not until the mid-1960's that a more formal theory of digital filtering began to emerge. By the late 1960's, digital filters have acquired a good deal of attention, owing to the advent of low-cost large-scale-integration (LSI) digital circuits and logic hardware, and the fact that digital filtering has distinct advantages over conventional analogue means. In fact, nowadays, digital filtering has been placed in a stance of growing importance due to its attractive processing power and the rapid advances in digital semiconductor technology.

Besides its application in seismology, digital filtering has also applications in other fields, for instance, in speech research and processing. In the early 1970's, L.R.Rabiner et.al. at the Bell Telephone Laboratories built a digital formant speech synthesizer which consisted of a number of digital filters (4). Since then the unit has been a satisfactory working example of the use of digital filtering. The advantages of this digital formant synthesizer over conventional analogue formant synthesizers include a precise control over centre frequencies and bandwidths of the resonators in the synthesizer, stability and reliability of the hardware, light weight, small size and low power consumption (4).

1.2 ADVANTAGES OF DIGITAL FILTERING:

The advantages of the digital formant synthesizer mentioned in the previous section are inherent in the use of digital filtering. In this section, we shall outline the main advantages of digital filtering as below:

(a) Programmability: The characteristics of a digital filter are entirely selected by the filter coefficients supplied to the filtering unit. Consequently, the same set of digital hardware in the filtering unit can be programmed into practically any type of filter characteristic while no other changes are required. Hence, the digital filter is, in this sense, a versatile and universal filtering unit. Furthermore, the filter coefficients are easily and readily implemented.

(b) Multiplexibility: In situations where the input bit rate (sampling rate times bits per sample) is significantly below the capability of the digital hardware used, a digital filter can be time-division-multiplexed to utilize the digital hardware more efficiently and cost-effectively. Additionally, digital multiplexers are readily implemented and much more economical than their analogue counterparts.

(c) Stability, Reliability and Repeatability: Provided a set of stable filter coefficients are supplied by a suitable design, the characteristics of a digital filter are completely stable, having no drift whatsoever under any circumstances (e.g. temperature variations, mechanical vibrations etc.). Thus, a digital filter is as reliable as the integrated circuits of which it is made. Since there is no drift, the performance of a stable digital filter is always repeatable.

(d) Accuracy: In general, digital filters can achieve frequency responses closer to ideal specifications than their analogue counterparts in many circumstances. The overall accuracy largely depends on the dynamic range of the filter variables and the word length of the filter coefficients. This can be increased at the expense of extra hardware. Thus, a tradeoff between accuracy and the resultant speed of operation achievable.

(e) Testability: Since a digital filter is in essence a logic unit,

it is very simple to test with a logic state analyzer. Also, debugging is simplified with the help of some built-in test programmes. Moreover, as there is no drift, there is no need for tuning, setting-up or alignment of components, and very little parametric testing. Thus, maintenance of a complex filtering unit becomes extremely straightforward and economical in comparison with many traditional analogue filtering units.

(f) Size and Ruggedness: In general, a digital filter has small size and light weight advantages over other systems. It is especially advantageous at very low frequencies when analogue components become very bulky. It is fully solid-state, and as such, it is extremely robust.

(g) Low-cost: This is normally a feature of a digital filter. As large-scale-integration (LSI) circuits are rapidly coming down in price, the cost of digital filtering will correspondingly be decreased, and in future, complicated digital filtering systems are likely to be available in LSI chips as standard off-the-shelf items.

In addition to the above, there is usually the benefit of low power consumption due to the multiplexibility of a digital filter as explained in (b) above and large-scale-integration of many digital hardware. Having mentioned the advantageous side of the use of digital filtering, we hasten to add that there are invariably certain snags in some circumstances where existing analogue means still apply or dominate.

1.3 DRAWBACKS OF DIGITAL FILTERING:

In this section, we shall outline some drawbacks in the use of digital filtering as follows:

(a) Operational Speed: Present day signal processing at microwave frequencies is pre-dominated if not exclusively performed by analogue means, though there has been recently reported some advances in microwave gigabit logic circuits (5),(6),(7). Thus, as far as signal processing at microwaves is concerned, digital filtering is at present very much handicapped in comparison with existing analogue means. Furthermore, going up in speed in digital filtering often involves parallel processing which implies

a corresponding increase in hardware cost, thereby rendering the use of digital filtering at high frequencies less attractive. In short, the operational speed of digital filters at present is very much limited and restricted by technology.

(b) Processing Bandwidth: A digital filtering unit capable of processing a signal with a large bandwidth can be multiplexed to process a bank of channels of smaller bandwidths. In this mode of operation, a high-speed filtering unit is cost-effectively employed to implement a large number of low-speed channels. However, when all the channels to be processed are wideband and operate at high speeds, comparable to that of the filtering unit, then the financial benefit of the multiplexibility of the use of digital filtering is lost. It is seen that the present issue is closely related to the operational speed of a digital filter as explained in (a) above.

(c) Coefficient Storage: If a large number of channels are required in a processing system and the filtering is to be implemented by means of a time-division-multiplex general-purpose filtering unit, then a correspondingly large amount of memory is needed to store the fixed filter coefficients, unless an algorithm is available to calculate the required coefficients in realtime. Therefore, in the former case, the cost of excessive memory requirement may be prohibitive, despite the fact that semiconductor memory is rapidly going down in price at present.

(d) Limitations of Analogue-to-Digital Converter: Due to present day technological limitations, the cost and speed of analogue-to-digital converters for high-frequency processing purposes have set a ceiling for the area of the applications of digital filtering. Moreover, in replacing an otherwise analogue system, additional costs in analogue-to-digital conversions and vice versa, plus all essential interfacing, may offset the previously mentioned advantages of the use of digital filtering. However, this situation will be changed in future when large-scale-integration (LSI) can produce high-speed analogue-to-digital converters at a relatively low cost. Hence, the present issue is technology-dependent.

1.4 OPTIMUM CONDITIONS FOR THE APPLICATION OF DIGITAL FILTERING:

In this section, we shall consider some optimum conditions (8) for the application of digital filtering. These conditions can be briefly summarized as follows:

(a) When the necessary analogue-to-digital conversion and/or digital-to-analogue conversion are already provided or are unnecessary as in the case of a pure digital environment.

(b) When a number of signals have to be filtered identically, thus allowing the processing hardware to be multiplexed.

(c) When filtering at extremely low frequencies where analogue components become bulky and inaccurate due to drift.

(d) When it is possible to simplify the processing task or reduce the cost of processing.

(e) When properties peculiar to digital filters enable systems to be implemented which cannot easily be realized with analogue means.

(f) When a general-purpose computer is already available so that a digital filter can be implemented via software. This often serves the purpose of simulation work and other off-line non-realtime applications, although on-line realtime tasks are sometimes feasible at very low frequencies.

1.5 OBJECTIVE OF THE THESIS:

Despite the vast amount of research in, and the growing importance of digital filters, digital filtering has not fully come of age, let alone achieving a widespread use or mass production. Until recent years, the implementation of digital filters has been constrained by technology to mainly applications off-line using the versatile general-purpose computer. High-speed special-purpose processors have been built in some circumstances for realtime signal filtering using expensive high-speed digital hardware which makes these processors and the application of digital filtering unattractive if not cost-prohibitive.

One of the reasons that contributes to the above mentioned situation is the lack of a suitable technology which is both fast and cost-effective. Another seemingly apparent reason will be the absence of an efficient design technique for realtime digital filtering at high speeds which can provide a reduction in the amount of computation

(and hence the volume of hardware) required. An equally important fact which has impaired the application of digital filtering is that existing approaches in the hardware implementation of digital filters have not been very successful in reducing the cost of digital filtering. Consequently, the main objective of this thesis is to propose a novel design technique and hardware implementation of digital filters for high-speed realtime filtering, in alleviation of the above mentioned situation.

In the proposed design technique, which is particularly suitable for designing filter banks, the use of a proposed frequency-sampling procedure results in a closed-form expression for a linear-phase equiripple passband structure which is further optimized by a proposed optimization procedure to yield a final optimum transfer function. Furthermore, the proposed design technique offers a reduction in the amount of computation (and hence the volume of hardware) required, even when implemented with existing approaches, in the implementation of digital filters. This is feasible via the modularity of the resultant design which allows both zero- and pole-sharing as well as realtime computations of filter coefficients, so as to reduce the storage requirement of, for instance, a filter bank implementation.

Aimed at cost reduction and speed improvement, the proposed hardware implementation of digital filters for high-speed realtime filtering does not include "adds" and "multiplies" of conventional arithmetic units. But instead, it employs a proposed algorithm, which we have called the "Carry Brought Forward Compensation Scheme" (CBFCS), in order to implement a proposed computational unit, which we have called the "Adderless-Multiplierless Unit" (AMU), to replace existing conventional arithmetic units, substituting table lookup for computation. Due to the rapidly increasing speed and capacity of semiconductor memory per unit of power dissipation, the proposed hardware implementation capitalizes on memory to offer significant reductions in cost and power consumption for the same speed of operation as that of existing approaches. In fact, the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) makes possible operational speeds which cannot be achieved by existing approaches by "sectioning" the carry propoagation path in the basic operation of

a conventional adder, which is the fundamental limitation in speed in all conventional processors.

Furthermore, a generalized quantization noise model is subsequently proposed for the analysis of finite-word-length effects on the above proposed hardware implementation. Additionally, a finite-word-length demonstration processor was built to verify experimentally the proposed hardware implementation and to further investigate the above generalized quantization noise model. The logic design of the above demonstration processor is at system level and the resultant architecture is very suitable for large-scale-integration (A suitable candidate would be CMOS on Sapphire which promises high volume yield and speed of operation). Finally, results from computer simulations and experiments with the finite-word-length demonstration processor showed a satisfactory agreement between theory and practice as far as the finite accuracy of the processor allowed.

1.6 PLAN OF THE THESIS:

A digital filter is first defined in the introductory chapter, followed by a brief historical background of digital filtering. Also, in this chapter, the advantages and drawbacks of the use of digital filtering are discussed thoroughly, thus leading to a set of optimum conditions for the application of digital filtering. The objective of the present thesis is also made clear at an early stage. Chapter two reviews the general theory and design techniques of existing digital filters. The relationship between analogue and digital filtering has been discussed, while comments on the above existing techniques have been given wherever appropriate. Chapter three discusses the proposed design technique of digital filters for high-speed filtering. The important concepts leading to the development of the proposed design technique have also been thoroughly discussed. Chapter four reviews existing hardware implementations of digital filters and their limitations. Chapter five introduces the proposed hardware implementation of digital filters for high-speed filtering which is meant to overcome the limitations of existing approaches. This chapter concludes with a critical comparison between the proposed hardware implementation and existing approaches with special reference to the cost, speed and power

consumption factors in the hardware implementation of digital filters. Chapter six introduces the effects of the use of finite word lengths on the proposed hardware implementation discussed in chapter five. A generalized quantization noise model is subsequently proposed for the error analysis of the proposed hardware implementation. This chapter also includes comparisons of the various aspects of noise performance between the proposed approach and existing approaches wherever appropriate. Chapter seven presents a logic design of a demonstration processor wherein lies a number of design aspects to which special techniques have been applied. This logic design is at system level so that a minimum number of logic gates have been used, and power consumption reduced. The resultant design has been biased towards a modular architecture which is very suitable for large-scale-integration. Chapter eight presents experimental results and simulation results of filters designed by the proposed design technique discussed in chapter three. The resultant noise performance is then compared with the predictions by the proposed generalized noise model discussed in chapter six. Since the above filters tested have been design by the proposed design technique, the results obtained in this chapter can then be compared with the theoretical predictions in chapter three. Chapter nine suggests some future work and applications of the proposed design technique and hardware implementation of digital filters for high-speed filtering while chapter ten concludes the present work.

CHAPTER TWO

THEORY AND DESIGN OF DIGITAL FILTERS

2.1 THEORY OF DIGITAL FILTERING: (9)

2.1.1 INTRODUCTION:

In the previous chapter, a digital filter has been defined as a computational process or an algorithm which can be implemented in computer software or digital hardware. The term digital implies that the time and amplitude variables of a digital filter are quantized. A digital filter therefore works in discrete-time with its signals represented as sequences of quantized numbers which can only take on a finite set of discrete values. These signals are normally digitized by some well defined rules or numeric representations, and are thus limited to a finite precision due to quantization. Figure 2.1.1 depicts the block diagram representation of a general digital filter.

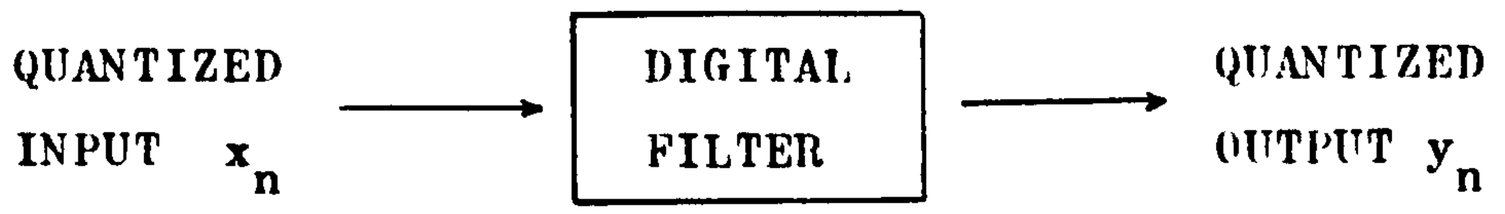


FIG: 2.1.1 BLOCK DIAGRAM REPRESENTATION
OF A GENERAL DIGITAL FILTER

In this section, we shall first discuss the theory of digital filtering as an analogue counterpart, prior to reviewing the techniques used in existing digital filter designs. Since the Z Transform plays a role in the discrete-time system theory analogous to that of the Laplace Transform in continuous-time system theory, its theory and derivation are given in appendix A.

2.1.2 DIGITAL FILTER AS AN ANALOGUE COUNTERPART: (10),(11),(12)

The concepts employed to describe, analyse and synthesize digital filters are analogous to those used in the analogue domain. For instance, a digital filter can be characterized by its impulse, frequency and phase responses in just the same way as an analogue filter is characterized by its input/output characteristics.

In the analogue domain, a linear filter is completely defined if either its impulse response h(t) or its frequency response H(s) is known. If an analogue signal x(t) is impressed on the input of the filter, the resultant output signal y(t) can be found from the convolution of x(t) and h(t) via the convolution integral in the time domain, viz.,

$$y(t) = \int_0^t x(t-\tau).h(\tau)d\tau \qquad (2.1.1)$$

The output spectrum Y(s) is given by the Laplace Transform as

$$Y(s) = \int_0^\infty y(t).e^{-st}dt \qquad (2.1.2)$$

Alternatively, if Y(s) is known, y(t) can be found by the Inverse Laplace Transform as

$$y(t) = \frac{1}{2\pi j}\oint Y(s).e^{st}ds \qquad (2.1.3)$$

where s is the Laplacian variable in the s-plane.

On the other hand, the frequency spectrum of the output signal Y(s) can be expressed as the product of the spectrum of the input signal X(s) and the transfer function H(s) of the filter. Conversely, if a particular output spectrum is desired in response to a given input, the filter response can be found from

$$H(s) = \frac{Y(s)}{X(s)} \qquad (2.1.4)$$

and can be synthesized from analogue components.

In the digital domain, the convolution equation relating the digital output y_n to the digital input x_n and impulse response h_n is simply a summation

$$y_n = \sum_{k=-\infty}^{\infty} x_{n-k}.h_k \qquad (2.1.5)$$

or

$$y(nT) = \sum_{k=-\infty}^{\infty} x((n-k)T).h(kT) \qquad (2.1.6)$$

when the sampling instants are equidistant.

The output digital spectrum is given by the Z Transform as

$$Y(z) = \sum_{n=o}^{\infty} y_n . z^{-n} \qquad (2.1.7)$$

Alternatively, if $Y(z)$ is known, y_n can be found by the Inverse Z Transform as

$$y_n = \frac{1}{2\pi j} \oint_c Y(z) . z^{n-1} dz \qquad (2.1.8)$$

where C is a closed contour within the regions of convergence of the power series in equation (2.1.7) and enclosing the origin. The complex variable z is known as the Standard Z Transform variable in the z-plane.

If the input spectrum $X(z)$ and the output spectrum $Y(z)$ are known, the frequency response of the digital filter is

$$H(z) = \frac{Y(z)}{X(z)} \qquad (2.1.9)$$

and can be synthesized using digital hardware.

In addition to the above, the analogue transfer function of an analogue filter is related to its impulse response by the transform pair

$$H(s) = \int_o^{\infty} h(t) . e^{-st} dt \qquad (2.1.10)$$

$$h(t) = \frac{1}{2\pi j} \oint H(s) . e^{st} ds \qquad (2.1.11)$$

and the digital transfer function of a digital filter being related to its impulse response by the transform pair

$$H(z) = \sum_{n=0}^{\infty} h_n . z^{-n} \qquad (2.1.12)$$

$$h_n = \frac{1}{2\pi j} \oint_c H(z) . z^{n-1} dz \qquad (2.1.13)$$

The operation of analogue filtering can also be described by a relevant differential equation,

$$y(t) = \sum_{k=o}^{\infty} a_k . \frac{\partial^k x(t)}{\partial t^k} + \sum_{k=1}^{\infty} b_k . \frac{\partial^k y(t)}{\partial t^k} \qquad (2.1.14)$$

while the analogue filter itself being a physical realization of the differential equation and the filter output is thus the solution of that differential equation when it is excited by an input signal. However, the discrete digital version of the continuous analogue differential equation is known as the difference equation, which becomes the analogue differential equation when the finite differences tend to zero i.e. the sampling rate tends to infinity. Thus the digital filter can be regarded as a physical realization of the difference equation with its output being the solution of the difference equation when excited by an input signal.

In general, a digital filter may be represented by a linear difference equation as follows:

$$y_n = \sum_{k=o}^{\infty} a_k . x_{n-k} + \sum_{k=1}^{\infty} b_k . y_{n-k} \qquad (2.1.15)$$

which states that the present output y_n is found by scaling and then adding the present input x_n and previous inputs x_{n-k} and outputs y_{n-k}. The order of the difference equation is said to be ∞, as in the case of the above analogue equation (2.1.14).

Furthermore, digital filters can be broadly divided into two types (a) on the basis of realization and (b) on the basis of their impulse responses. When describing digital filters on the basis of their realizations, the following terms are used:

(i) A recursive filter is a discrete-time filter which is realized via a recursion relation i.e. the output samples of the filter are explicitly determined as a weighted sum of the past output samples as well as past and/or present input samples. For instance, via the feedback relationship

$$y_n = a_o . x_n + a_1 . x_{n-1} + a_2 . x_{n-2} - b_1 . y_{n-1} - b_2 . y_{n-2} \qquad (2.1.16)$$

with a transfer function $H(z)$ represented as

$$H(z) = \frac{a_2 . z^{-2} + a_1 . z^{-1} + a_o}{b_2 . z^{-2} + b_1 . z^{-1} + 1} \qquad (2.1.17)$$

(ii) A nonrecursive filter is a discrete-time filter for which the output samples of the filter are explicitly determined as a weighted sum of past and present input samples only. For instance, the nonrecursive relationship

$$y_n = a_o.x_n + a_1.x_{n-1} + a_2.x_{n-2} \qquad (2.1.18)$$

A nonrecursive filter is also known as a transversal filter.

The transfer function for the above second-order difference equation is

$$H(z) = a_o + a_1.z^{-1} + a_2.z^{-2} \qquad (2.1.19)$$

When describing digital filters on the basis of their impulse responses, the following terms are used:

(i) A finite impulse response (FIR) filter is a filter whose impulse response h_n is zero outside some finite limits i.e., $h_n=0$, for $n>N_1$ and $n<N_2$ with $N_1 \geqslant N_2$.

(ii) An infinite impulse response (IIR) filter is a filter for which either $N_1=\infty$ or $N_2=-\infty$ or both in (i). Thus the duration of the impulse response of the digital filter is infinite.

It should be noted that the terms recursive and nonrecursive are recommended as descriptions for how a filter is realized and not whether or not the impulse response of the filter is of finite duration. Although IIR filters are generally realized recursively and FIR filters are generally realized nonrecursively, IIR filters can be realized nonrecursively while FIR filters can also be realized recursively.

2.1.3 RELATIONSHIP BETWEEN DISCRETE AND CONTINUOUS SIGNALS:(9)—(13)

If a sequence arises as a result of periodic sampling of a continuous-time signal $x(t)$, that is $x(nT)=x_n$ where T is the sampling period, then $X(z)$, the Z Transform of $x(t)$, is related to $X(s)$, the Laplace Transform of $x(t)$ by the relationship

$$X(z)\Big|_{z=e^{sT}} = \sum_{n=-\infty}^{\infty} x(nT).e^{-snT}$$

$$= \frac{1}{T}\sum_{k=-\infty}^{\infty} X\left(s+j.\frac{2\pi k}{T}\right) \qquad (2.1.20)$$

The above relationship can be arrived at by considering the ideal sampling process by an impulse train. The ideal impulse sampler is shown in figure 2.1.2 below:

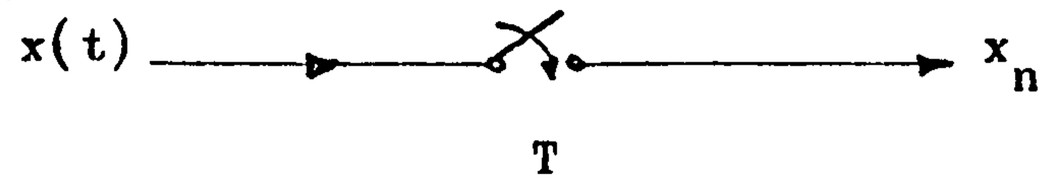


FIG: 2.1.2 IDEAL IMPULSE SAMPLER

The sampling function,

$$s(t) = \sum_{k=-\infty}^{\infty} \delta(t-kT) \qquad (2.1.21)$$

is a periodic train with a Fourier series representation

$$s(t) = \sum_{k=-\infty}^{\infty} C_k . e^{jk\omega_s t} \qquad (2.1.22)$$

where $\omega_s = 2\pi/_T$ is the sampling rate, while

$$C_k = \frac{1}{T}\int_{-T/2}^{T/2} \delta(t) . e^{-jk\omega_s t}\, dt$$

$$= \frac{1}{T}$$

Therefore, $$s(t) = \frac{1}{T}\sum_{k=-\infty}^{\infty} e^{jk\omega_s t}$$

and, $$X(z) = \mathcal{L}\left[\frac{1}{T}\sum_{k=-\infty}^{\infty} e^{jk\omega_s t} . x(t)\right]$$

$$= \frac{1}{T}\sum_{k=-\infty}^{\infty} \mathcal{L}\left[x(t) . e^{jk\omega_s t}\right]$$

$$= \frac{1}{T}\sum_{k=-\infty}^{\infty} X\left(s + j . \frac{2\pi k}{T}\right)$$

which is equation (2.1.20).

This relationship implies that the s-plane and the z-plane are inter-related in a many-to-one manner and not by a one-to-one mapping. The s-plane is seen to be divided into an infinite number of horizontal strips of width $2\pi/_T$, each of which maps into the entire z-plane under the mapping $z=e^{sT}$. The contributions

from individual strips are added together to produce X(z) as shown in figure 2.1.3 below:

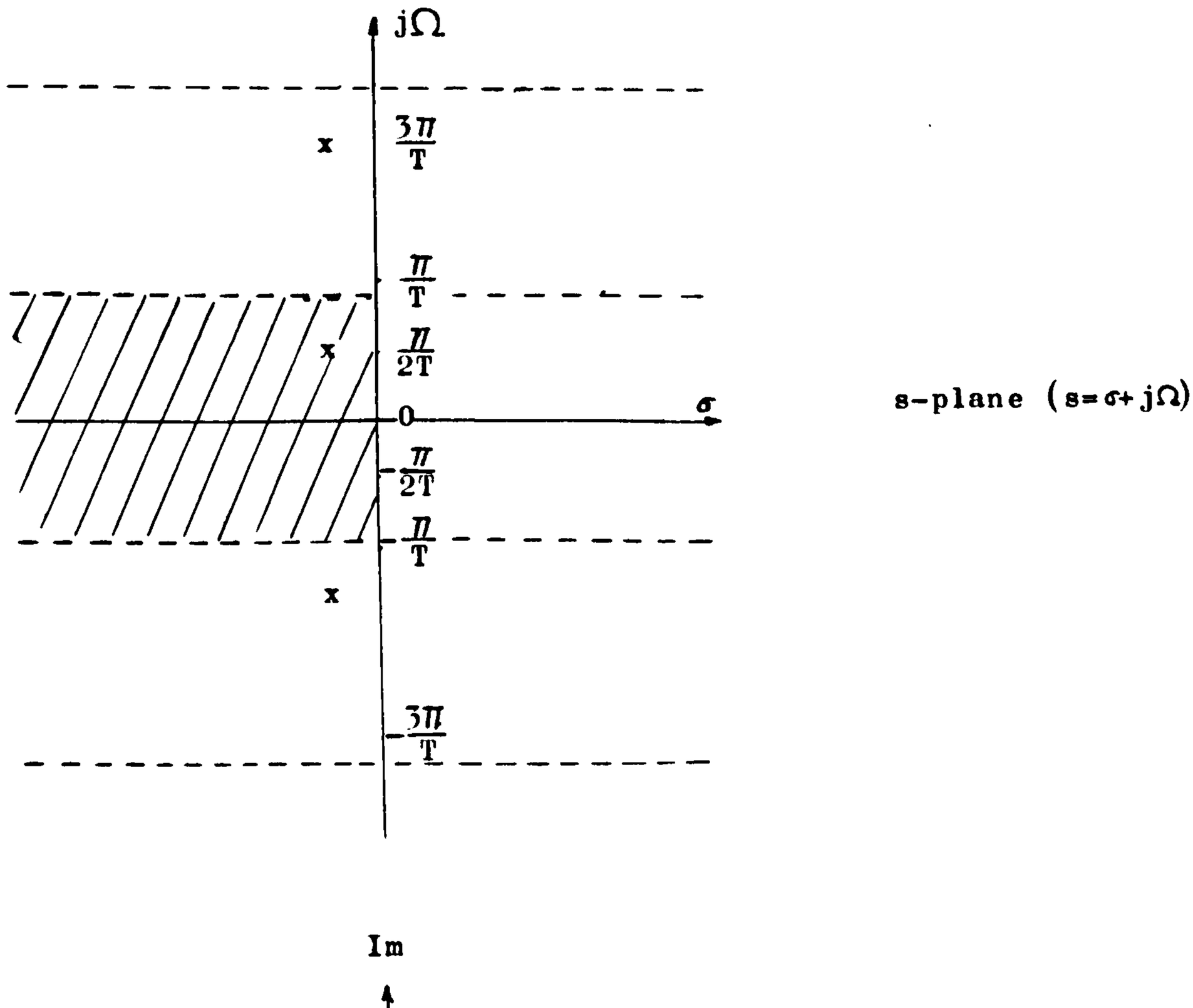


FIG: 2.1.3 RELATIONSHIP OF THE S-PLANE AND THE Z-PLANE AS SHOWN BY THE MAPPING $z=e^{sT}$.

In the above figure, the analogue frequency variable has been designated Ω in the s-plane while the digital frequency variable

has been designated ω in the z-plane, to conform with notations adopted in current literature in digital filtering. Furthermore, the point $z=1+j0$ is often casually referred to as the d.c. point since it corresponds to the point $s=0$ in the s-plane.

It is further noticed that the left half of each s-plane strip maps into the interior of the unit circle, while the right half of each s-plane strip maps into the exterior of the unit circle. The imaginary axis of the s-plane maps onto the unit circle in the z-plane in such a manner that each segment of length $2\pi/_T$ is mapped once around the unit circle which has the equation

$$e^{j\omega T} = \cos\omega T + \sin\omega T \qquad (2.1.23)$$

For instance, the points $\Omega = \pm \pi/_{2T}$ on the $j\Omega$-axis are mapped onto the points $z=0\pm j1$ on the unit circle, while the point $\Omega = \pi/_T$ is mapped onto the point $z=-1+j0$ in the z-plane.

One significant difference between continuous-time analogue and discrete-time digital systems arises here — digital systems have pole-zero patterns which are periodic in frequency along the ω-axis as indicated by equation (2.1.20). The repetition of the analogue spectrum $X(s)$ is a result of the essential sampled-data representation of the analogue signal $x(t)$, and occurs at intervals of $1/_T$ hertz where T is the sampling interval.

Finally, to conclude this section, it is worth pointing out that a link between continuous-time analogue and discrete-time digital systems is provided by the Sampling Theorem which is the topic of interest in the next section.

2.1.4 THE SAMPLING THEOREM:

From the discussion in the previous section, we notice that sampling a continuous-time analogue signal $x(t)$ introduces the effect of "Frequency Aliasing" as depicted in figure 2.1.4. This effect is predicted by the Sampling Theorem which states that:

(a) If an analogue signal $x(t)$ contains no frequency higher than Ω, it is then completely determined by the values of its magnitudes at a series of sampling instants less than $\pi/_\Omega$ seconds apart, where $\omega_s \geqslant 2\Omega$ is the sampling frequency and T is the period. This will apply to analogue signals containing frequencies in the range from 0 to Ω.

(b) If the analogue frequency spectrum X(s) being sampled does not begin from zero frequency, then a more general form of the Sampling Theorem will apply. This states that if an analogue signal x(t) is bandlimited to a bandpass spectrum which extends from Ω_1 to Ω_2 in frequency, then the minimum sampling frequency required is $2\Omega_2/_m$ where m is the largest integer not to exceed $\frac{\Omega_2}{\Omega_2-\Omega_1}$.

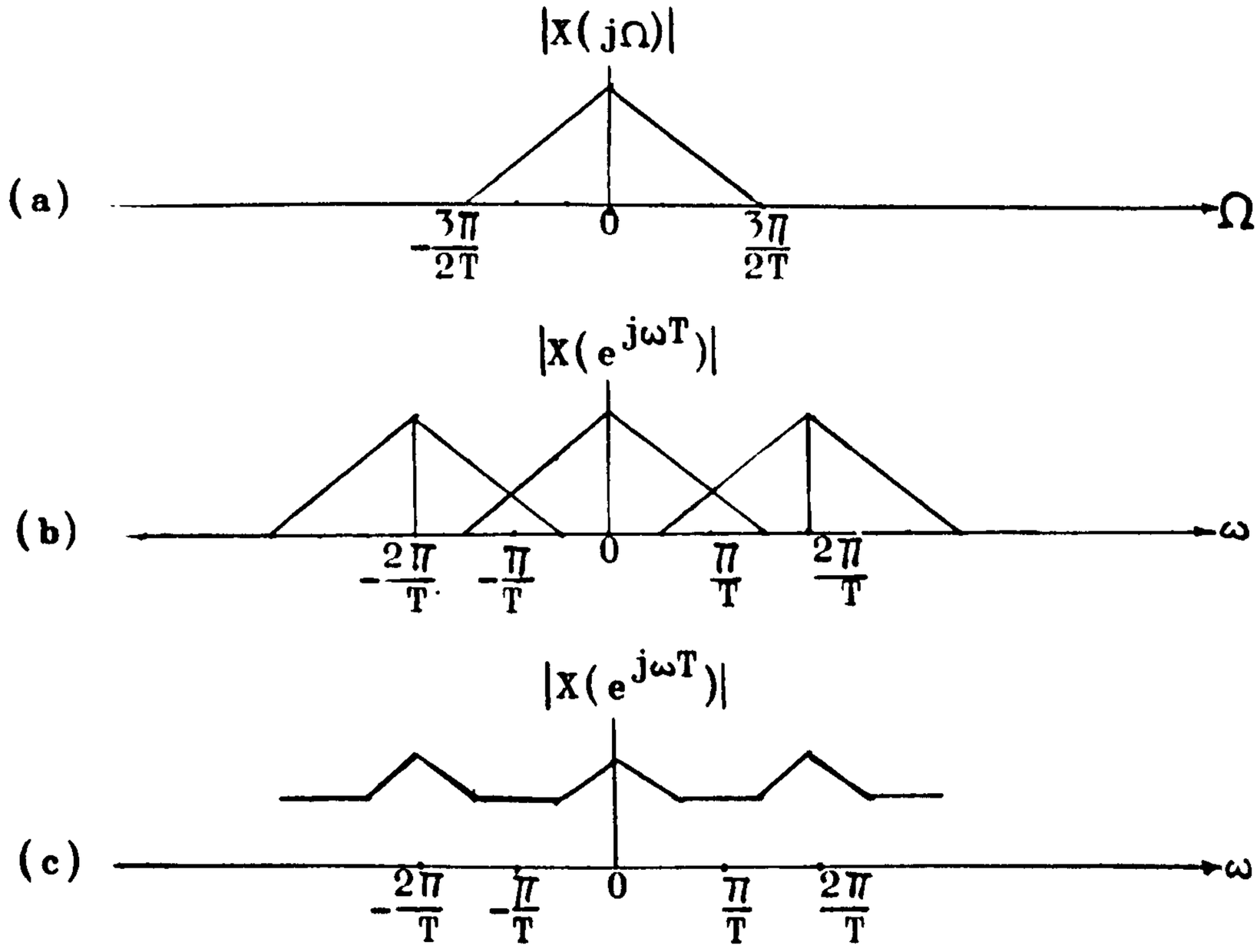


FIG: 2.1.4 EFFECT OF FREQUENCY ALIASING ON THE DIGITAL FREQUENCY SPECTRUM

Figure 2.1.4a shows the spectrum of an analogue signal which is sampled at a frequency $\omega_s=2\pi/_T$ as shown in figure 2.1.4b. The resultant sampled digital spectrum is an aliased version of the original analogue spectrum as shown in figure 2.1.4c. Therefore, in view of the above mentioned facts, it is essential to bandlimit the analogue signal x(t) by a sharp cutoff lowpass filter with a cutoff frequency at Ω_c, where Ω_c is less than or equal to half the sampling frequency ω_s, before sampling it to form x_n, the equivalent digital sequence which will then be alias-free and preserving all the information contained in x(t). This is depicted in figure 2.1.5 where the sampling frequency is set at $\omega_s=2\Omega_c$.

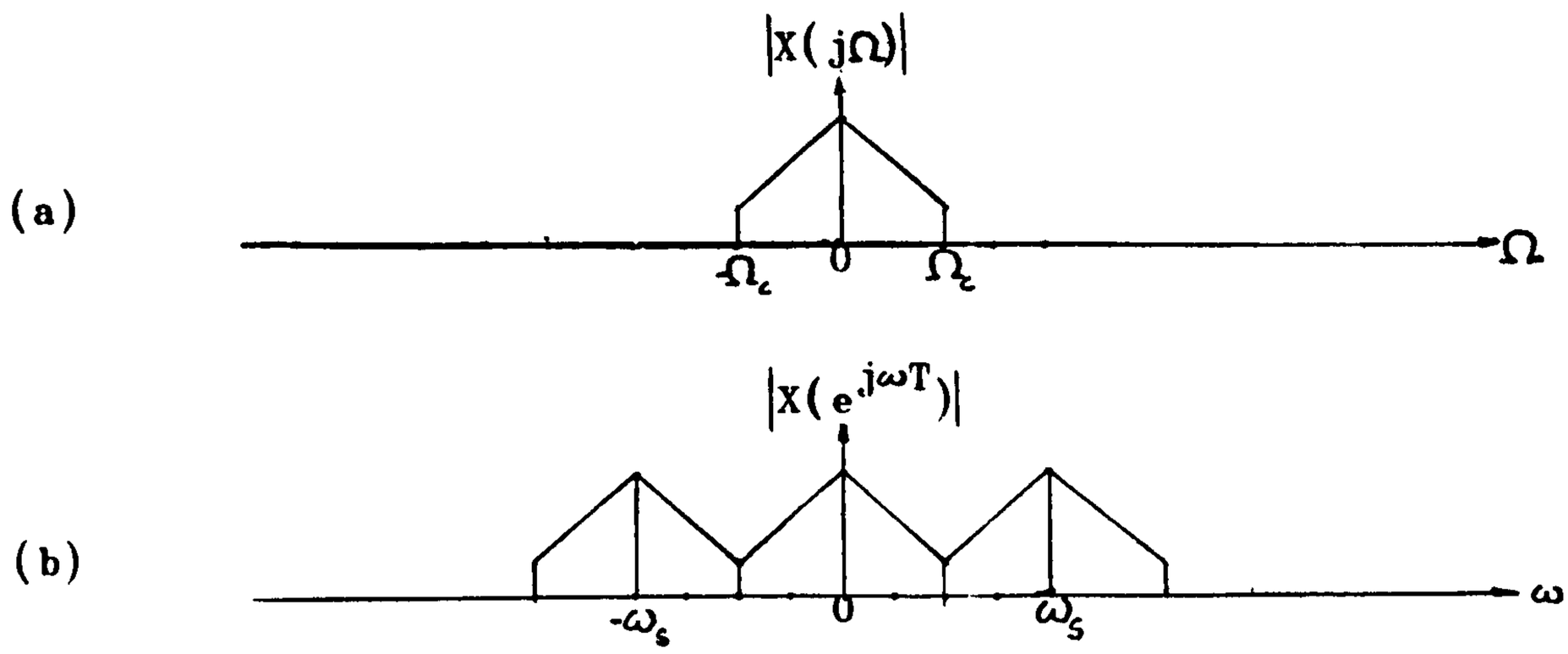


FIG: 2.1.5 SAMPLED SPECTRUM OF AN IDEAL BANDLIMITED ANALOGUE SIGNAL WITH SAMPLING RATE $\omega_s = 2\Omega_c$

Figure 2.1.5a shows the ideal bandlimited analogue spectrum and figure 2.1.5b shows the alias-free sampled spectrum. The repetitive nature of the resultant digital spectrum $X(e^{j\omega T})$ is obvious in figures 2.1.4 and 2.1.5. This is fundamental in all digital filter designs. In the next section, we shall review and comment on existing design philosophies of digital filters.

2.2 A REVIEW OF EXISTING DESIGN PHILOSOPHIES OF DIGITAL FILTERS:

An important class of techniques for designing digital filters from analogue filters is based on a transformation defined by:

$$H(s) = \frac{\sum_{i=0}^{M} a_i . s^i}{\sum_{i=1}^{N} b_i . s^i} = \frac{\prod_{i=1}^{M} (s+c_i)}{\prod_{i=1}^{N} (s+d_i)} \Longrightarrow H(z) = \frac{\sum_{i=0}^{M} a_i . z^{-i}}{1+\sum_{i=1}^{N} b_i . z^{-i}} \qquad (2.2.1)$$

As a matter of fact, digital filters with Infinite Impulse Response (IIR) to be realized recursively, have invariably been designed via a mapping from the analogue domain to the digital domain. In our discussion, we shall begin with techniques of digitizing continuous-time analogue filters before devoting the rest of the chapter to existing direct designs of digital filters. Design techniques for Finite Impulse Response (FIR) filters will also be dealt with

accordingly, before giving a general comment on all techniques in the conclusion.

2.2.1 IMPULSE INVARIANCE TRANSFORMATION:(10),(11),(14)

This transformation for digitizing an analogue filter is also known as the Standard Z Transform in which the impulse response of the derived digital filter is identical to the sampled impulse response of the continuous-time filter. Thus the nature of the Impulse Invariance Transformation is represented by:

$$h_n = h(t)\Big|_{t=nT}$$

$$= h(nT) \qquad (2.2.2)$$

for all values of n . If the analogue filter has a transfer function given by:

$$H(s) = \frac{\sum_{k=o}^{M} d_k . s^k}{\sum_{k=o}^{N} c_k . s^k} = \sum_{k=1}^{N} \frac{A_k}{s+s_k}$$

then the requirement in equation (2.2.2) implies that the derived digital spectrum H(z) is obtained from the partial fraction expansion of H(s) by the substitution:

$$\frac{1}{s+s_k} \longrightarrow \frac{1}{1- e^{-s_k T} . z^{-1}} \qquad (2.2.3)$$

such that,

$$H(s) = \sum_{k=1}^{N} \frac{A_k}{s+s_k} \longrightarrow H(z) = \sum_{k=1}^{N} \frac{A_k}{1- e^{-s_k T} . z^{-1}} \qquad (2.2.4)$$

Under the above transformation, we observe that a pole at $s=s_k$ in the s-plane is mapped to a pole at $e^{s_k T}$ in the z-plane and the coefficients in the partial fraction expansion of H(s) and H(z) are equal. If the analogue filter is stable, corresponding to the real part of s_k less than zero, then the magnitude of $e^{s_k T}$ will be

less than unity, so that the corresponding pole in the digital filter is inside the unity circle, which is the criterion of a stable digital system. Therefore, a stable analogue filter is then mapped into a stable digital filter with the impulse response preserved under the transformation. Furthermore, the zeros of the derived digital filter, in particular, are a function of the poles and the coefficients A_k in the partial fraction expansion of $H(s)$ and they will not in general be mapped by the relationship $z=e^{sT}$ as the poles are mapped.

One disadvantage of the Impulse Invariance Transformation is its inherent aliasing nature by virtue of the sampling process, such that $H(s)$ and $H(z)$ are related by the equation below:

$$H(z)\Big|_{z=e^{sT}} = \frac{1}{T}\sum_{k=-\infty}^{\infty} H\left(s+j.\frac{2\pi k}{T}\right) \qquad (2.2.5)$$

The frequency response of the digital filter is related to that of the analogue filter as

$$H(e^{j\omega T}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} H\left(j\Omega + j.\frac{2\pi k}{T}\right) \qquad (2.2.6)$$

From our previous discussion of the Sampling Theorem, it is clear that if and only if

$$H(j\Omega) = 0 \qquad \frac{\pi}{T} \leqslant |\Omega| \qquad (2.2.7)$$

then,

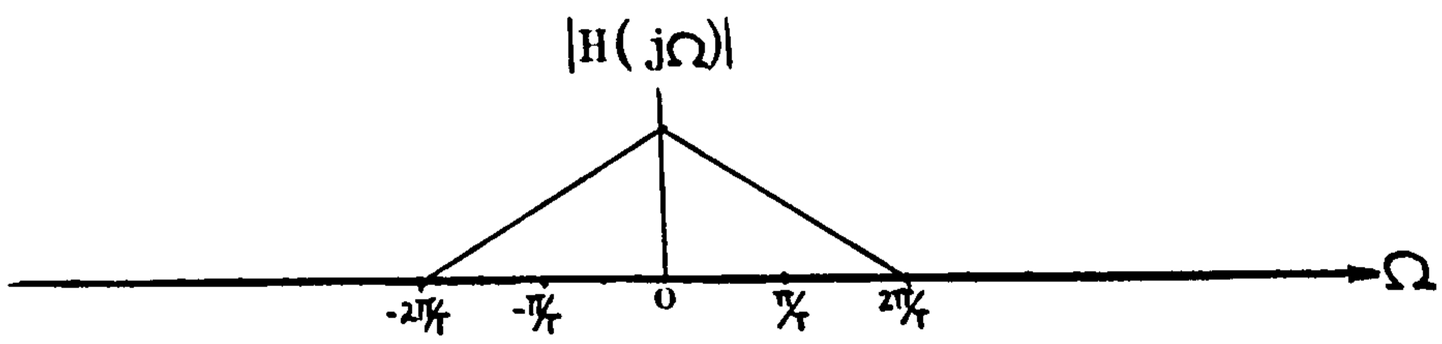
$$H(e^{j\omega T}) = \frac{1}{T}.H(j\Omega) \qquad |\omega T| \leqslant \pi \qquad (2.2.8)$$

Unfortunately, any practical analogue filter will not be bandlimited, and consequently, there is interference between successive terms in equation (2.2.6) due to aliasing as depicted in figure 2.2.1b. From the relationship $z=e^{sT}$ it is seen that strips of width $2\pi/_T$ in the s-plane are mapped into the entire z-plane as previously depicted in figure 2.1.3. From equation (2.2.5) above, it is clear that each horizontal strip of the s-plane is overlaid onto the z-plane to form the digital system function from the analogue system function. Thus, the Impulse Invariance Transform does not correspond to a simple algebraic mapping of the s-plane to the z-plane. In fact, as previously pointed out, the s-plane is being

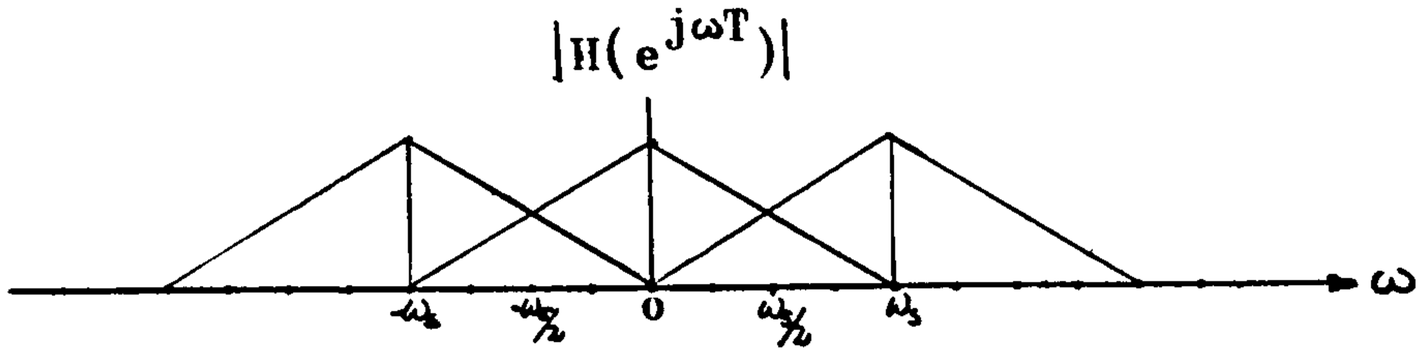
mapped into the z-plane by the many-to-one mapping $z=e^{sT}$. In fact, the poles shown in fig. 2.1.3, seperated by $2\pi/_T$ in frequency, are all mapped onto the same position in the z-plane.

In summary, the basis for Impulse Invariance as described above is to choose a unit sample response for the digital filter that is similar in some sense to the impulse response of the analogue filter. The use of this procedure is often not motivated so much by the desire to maintain the impulse response shape, but by the knowledge that if the analogue filter is bandlimited, then the derived digital filter frequency response will closely approximate the analogue frequency response. Although in the Impulse Invariance design procedure, distortion in the digital frequency response is introduced due to aliasing, the relationship between analogue and digital frequencies is linear. Consequently, except for aliasing, the shape of the analogue frequency response is preserved. If aliasing effects are negligible, H(s) and H(z) are simply related by equation (2.2.8). Therefore, the Impulse Invariance technique is obviously only suitable for narrowband or essentially bandlimited filter designs. For wideband designs where the passbands of the digital filters are an appreciable fraction of the Nyquist Interval (i.e. approximately 80% of the range $-\omega_s/_2 \leqslant \omega \leqslant \omega_s/_2$), additional bandlimiting, such as a "Guard Filter", is then required to avoid severe aliasing distortion. The same will apply to highpass and bandstop filter designs using the above technique. The Impulse Invariance Transformation is then applied to the cascade combination of the guard filter G(s) and the analogue filter H(s), where the guard filter G(s) is used to bandlimit H(s). The guard filter is actually a sharp cutoff lowpass filter. The order of the resultant digital filter thus obtained will be higher and is equal to that of the lowpass guard filter G(s) plus that of the analogue filter H(s) from which the digital filter is derived. Fig. 2.2.1 shows the effect of a non-bandlimited H(s) before and after the cascading of a guard filter G(s) under the Impulse Invariance Transformation. The sampling of a frequency spectrum is also illustrated. Fig.2.2.1a and 2.2.1b show the alinsing effect of the derived digital filter from a non-bandlimited analogue filter H(s) without a guard filter while fig. 2.2.1c and 2.2.1d clearly show the bandlimiting effect of the insertion of a sharp cutoff

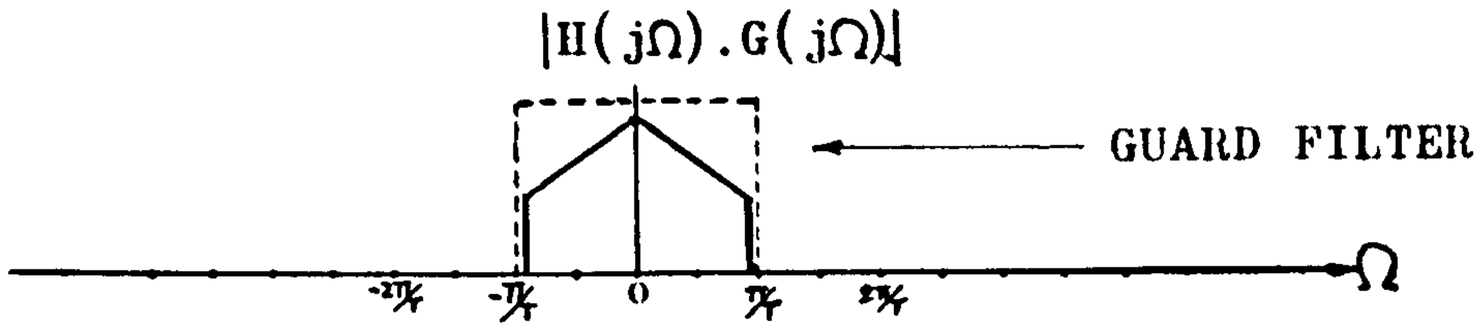
guard filter G(s) to produce an alias-free digital filter under the Impulse Invariance Transformation.



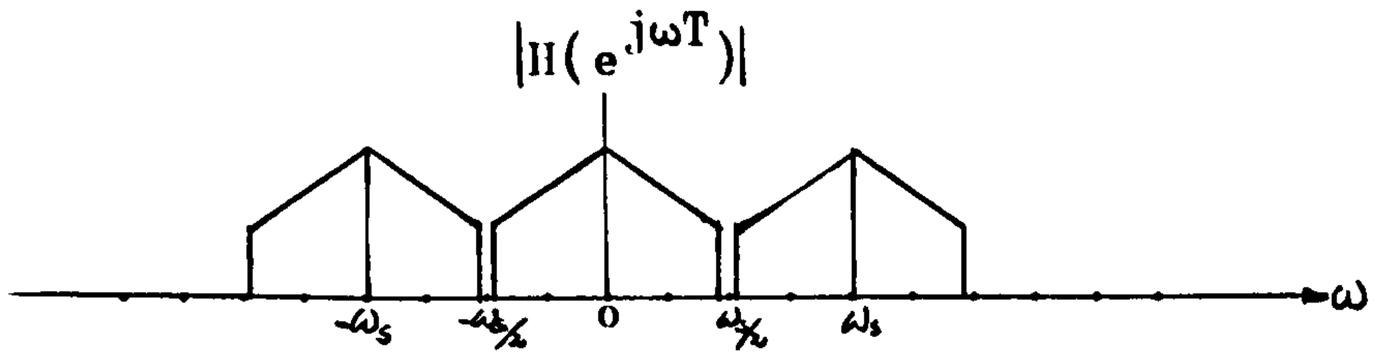
(a) ANALOGUE FILTER SPECTRUM UNBANDLIMITED BEFORE SAMPLING



(b) ALIASED DIGITAL FILTER SPECTRUM AFTER SAMPLING



(c) BANDLIMITED ANALOGUE FILTER SPECTRUM BEFORE SAMPLING



(d) ALIAS-FREE DIGITAL FILTER SPECTRUM AFTER SAMPLING

FIG: 2.2.1 DIGITAL FILTERS DERIVED FROM AN UNBANDLIMITED ANALOGUE FILTER H(s) WITHOUT AND WITH A GUARD FILTER G(s) BY THE IMPULSE INVARIANCE DESIGN.

In addition to the above mentioned, digital filters derived by the Impulse Invariance Transform have a gain approximately $1/_T$ times that of the analogue filter, due to the $1/_T$ multiplier in equation 2.2.5. This is generally compensated by multiplying each factor in equation 2.2.4 by T, so that the gain of the digital filter will be approximately the same as that of the analogue filter. Thus equation 2.2.3 is accordingly modified as

$$\frac{1}{s + s_k} \longrightarrow \frac{T}{1 - e^{-s\,T}.z^{-1}} \qquad (2.2.9)$$

2.2.2 BILINEAR Z TRANSFORM: (10),(11),(14),(15),(19)

This is also known as the s_1-plane Transform or Z-form which is a technique used to circumvent the aliasing problem of the Impulse Invariance Transform due to the combined contributions of all the horizontal strips in the s-plane shown before.

This transformation first maps the entire analogue s-plane into horizontal strips in an intermediate digital s_1-plane bounded by the lines $s_1=j(r-\frac{1}{2})\omega_s$ and $s_1=j(r+\frac{1}{2})\omega_s$ where ω_s is the sampling frequency and r is an integer. The intermediate s_1-plane is then mapped into the final z-plane by the relationship $z=e^{s_1T}$. Therefore, the Bilinear Z Transform can be viewed as a two-step process which uses an algebraic transformation to squash the entire $j\Omega$-axis into intervals of $2\pi/T$ wide on the $j\omega$-axis in the s_1-plane defined by the digital frequency variable ω in the z-plane. The above algebraic manipulation is given by

$$s = \frac{2}{T}.\tanh\left(\frac{s_1T}{2}\right) \qquad (2.2.10)$$

where T is the sampling period.

Since the hyperbolic tangent function is a monotonic function it follows that all the values of s are preserved in their correct order, but, because of the periodic nature of the relationship, the s-plane is mapped onto the s_1-plane in a series of parallel strips as shown below.

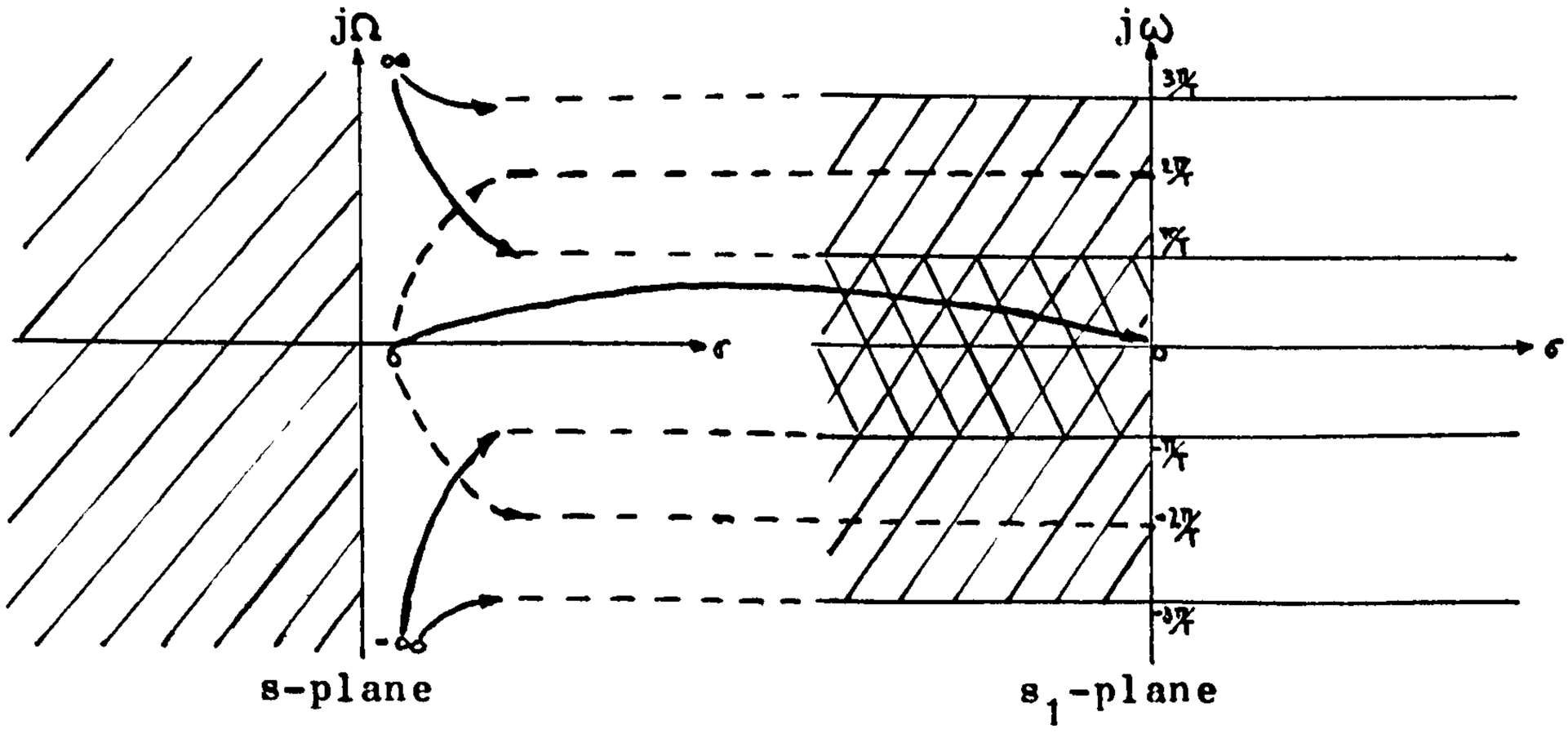


FIG: 2.2.2 MAPPING OF THE S-PLANE TO THE S_1-PLANE

The doubly shaded region in fig. 2.2.2, situated around the origin, is called the "baseband region" in which the baseband digital filter spectrum lies. From equation(2.2.10), we have,

$$\Omega = \frac{2}{T}.\tan\left(\frac{\omega T}{2}\right) \qquad (2.2.11)$$

so that the entire $j\Omega$-axis is mapped monotonically in the regions $(2r-1)\pi/_T < \omega < (2r+1)\pi/_T$, for all integer values of r, onto the $j\omega$-axis in the s_1-plane. On substitution of $z=e^{s_1T}$ in equation (2.2.10), we have the algebraic transformation of the analogue s-plane to the digital z-plane directly as:

$$s = \frac{2}{T}.\frac{1-z^{-1}}{1+z^{-1}} \qquad (2.2.12)$$

Thus, in terms of the z^{-1} variable in the z-plane, this transformation has the effect of mapping the entire s-plane onto the z-plane in such a way that the left half of the s-plane maps completely into the interior of the unit circle, and the right half of the s-plane maps into the exterior of the unit circle in the z-plane. The entire $j\Omega$-axis in the s-plane is mapped monotonically onto the unit circle in the z-plane. Therefore, the inherent aliasing errors in the Impulse Invariance Transformation are thus eliminated, since no folding occurs. These properties are easily shown by solving for z in terms of s in equation(2.2.12) as below:

$$z = \frac{\left(\frac{2}{T}\right) + s}{\left(\frac{2}{T}\right) - s} \qquad (2.2.13)$$

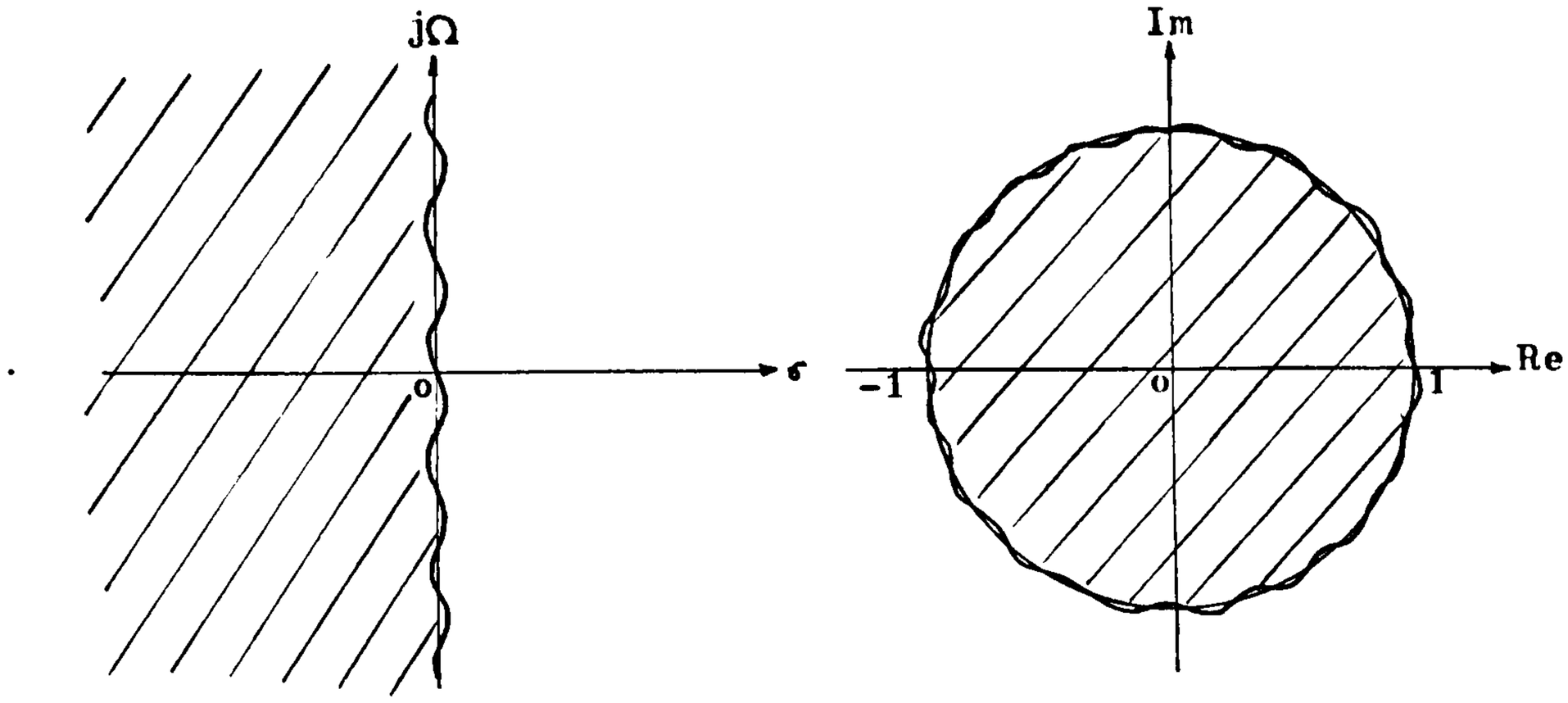
When $s=j\Omega$

$$z = \frac{\left(\frac{2}{T}\right) + j\Omega}{\left(\frac{2}{T}\right) - j\Omega}$$

therefore, $|z| = 1$

which is the equation of the unit circle.

The nature of the mapping $z=e^{s_1T}$ for the s_1-plane to the z-plane is the same as $z=e^{sT}$ shown in fig. 2.1.3, except for a change in variable from the analogue variable s to the digital variable s_1. Fig. 2.2.3 depicts the Bilinear Transformation.



s-plane z-plane

FIG: 2.2.3 THE BILINEAR TRANSFORMATION.

Under the Bilinear Z Transform, we have, $z=1$ when $\Omega=0$ and $z=-1$ when $\Omega=\infty$. Between these limits, the angle of $z=e^{j\omega T}$ varies monotonically from 0 to π. Equation(2.2.11) can be rewritten as:

$$\frac{\Omega T}{2} = \tan\left(\frac{\omega T}{2}\right) \qquad (2.2.14)$$

and it is seen that there is a highly non-linear relationship, however, between the analogue frequency Ω and the digital frequency ω. The nature of the warping or distortion of the frequency scale makes the Bilinear Z Transform unsuitable for systems whose frequency responses are not piewise constant; for example, an analogue differentiator. In addition to distortions in frequency responses, the Bilinear Z Transform in general, does not preserve impulse and phase responses of the derived digital filters. Fig. 2.2.4 shows the relationship between the analogue and digital frequency scales due to equation(2.2.11).

Despite its frequency warping effect, the Bilinear Transform can be used for digital systems whose frequency responses are piecewise constant; for example, a bandpass or bandstop filter etc. The above adverse effect of frequency warping is then compensated by a process called "Frequency Pre-warping" which is shown in fig. 2.2.5.

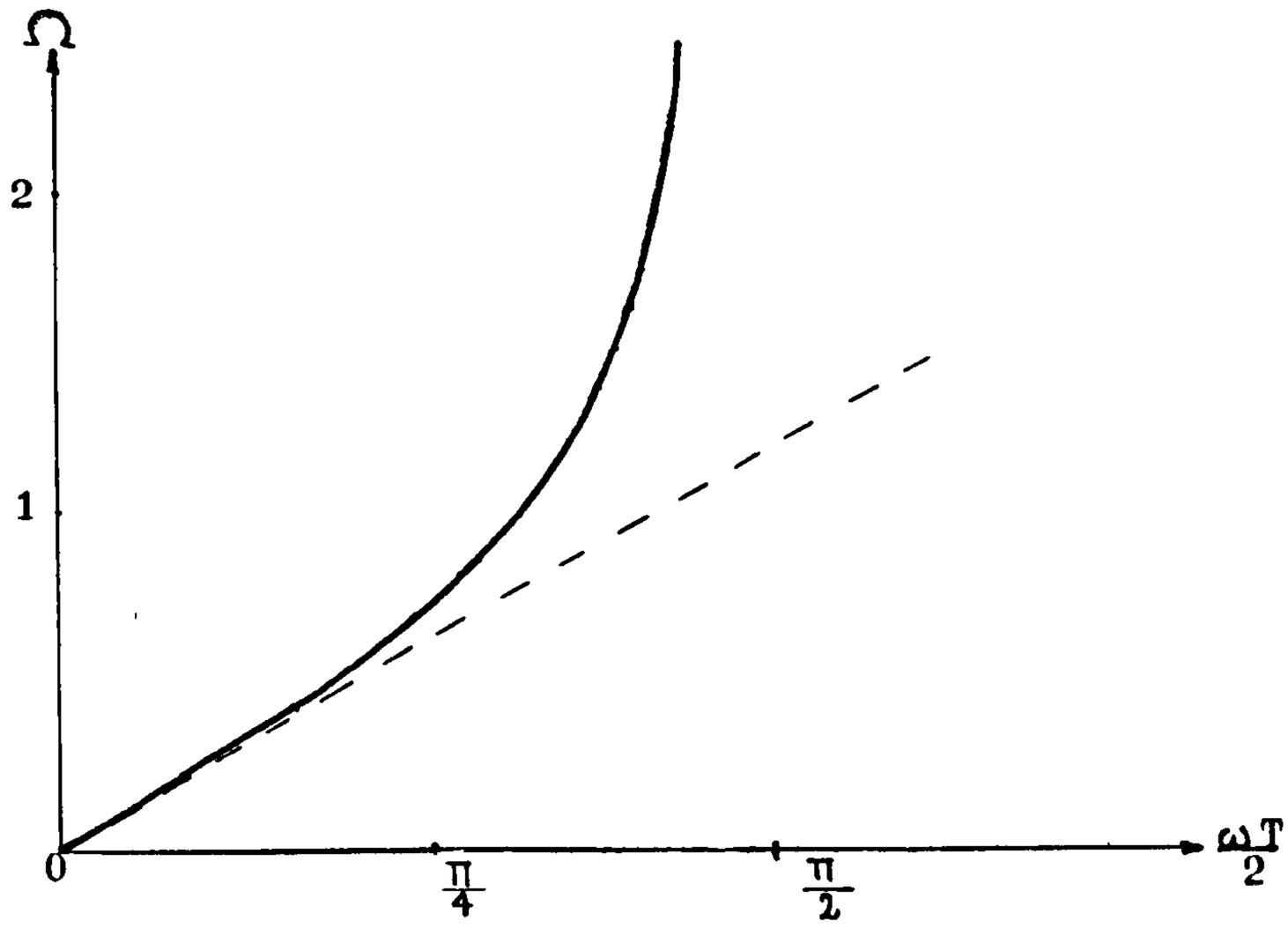


FIG: 2.2.4 RELATIONSHIP BETWEEN ANALOGUE AND DIGITAL FREQUENCY SCALES UNDER THE BILINEAR Z TRANSFORMATION

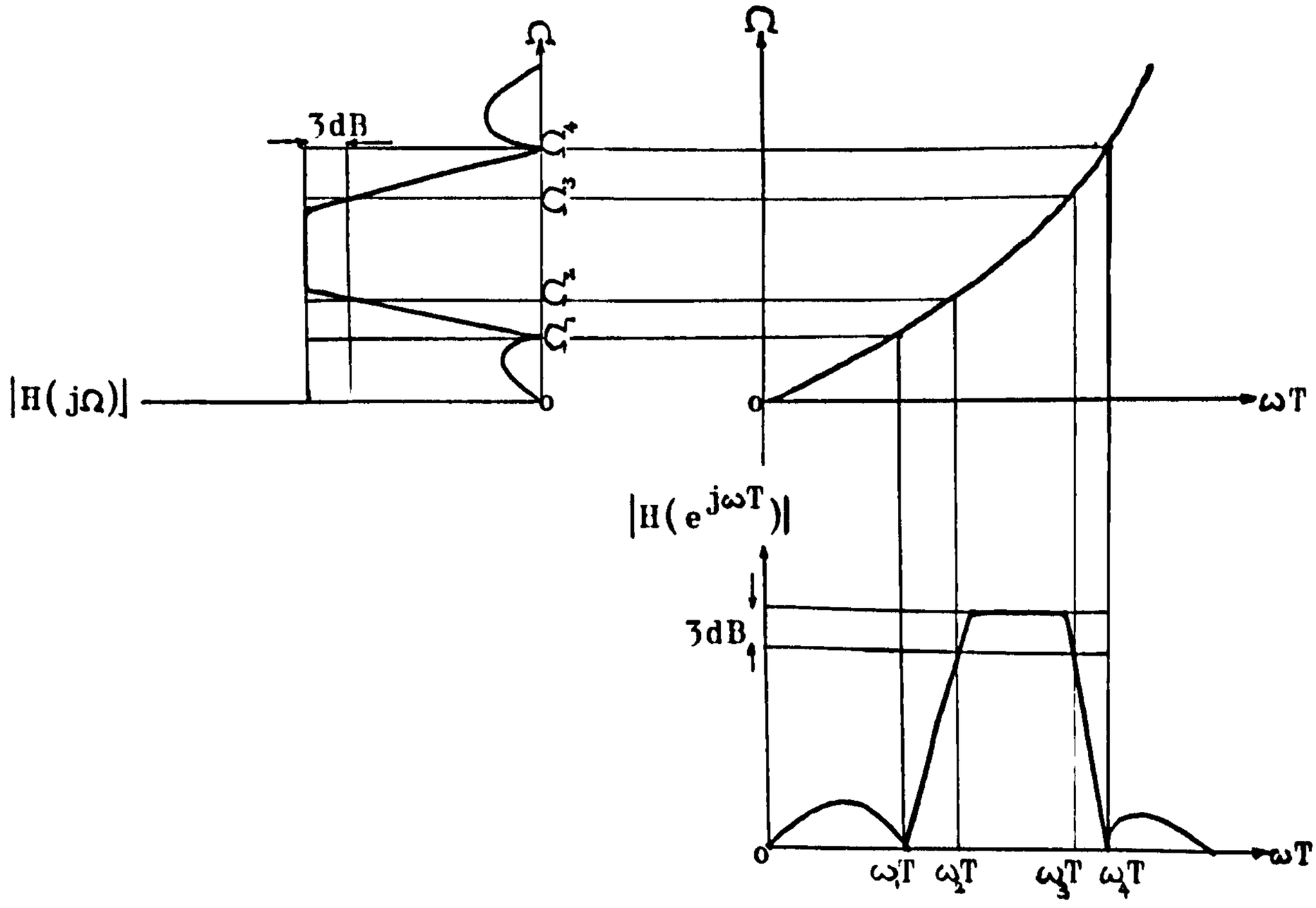


FIG: 2.2.5 TECHNIQUE FOR COMPENSATING THE NON-LINEAR FREQUENCY WARPING OF THE BILINEAR TRANSFORM

The desired critical set of digital filter cutoff frequencies is determined, as shown at the lower right-hand corner of fig.2.2.5. In fig. 2.2.5 we assume that there are four such frequencies(ω_1, ω_2, ω_3, ω_4). Using the frequency warping relation i.e. equation(2.2.11) between the analogue and digital frequency scales, the digital filter cutoff frequencies are now converted to a new set of analogue cutoff frequencies (Ω_1, Ω_2, Ω_3, Ω_4) as shown at the upper left-hand corner of fig.2.2.5. Finally an analogue filter is designed with the appropriate prewarped cutoff frequencies. Applying the Bilinear Z Transformation to this analogue filter gives a digital filter with the desired cutoff frequencies above.

Although the Bilinear Z Transformation maps a stable analogue filter H(s) into a stable digital filter H(z), the phase response of H(s) is not preserved due to a warping of the frequency scale, and the phase response of H(z) will be a warped version of the analogue phase response. The envelope or group delay of the analogue filter is defined as

$$\tau(\Omega) = -\frac{d}{d\Omega}\angle H(j\Omega) \qquad (2.2.15)$$

Similarly, the group delay of the digital filter is

$$\phi(\omega) = -\frac{d}{d\omega}\angle H(e^{j\omega T}) \qquad (2.2.16)$$

It follows from equation(2.2.11) that the analogue and digital phase responses are related as

$$\angle H(e^{j\omega T}) = \angle H(j\frac{2}{T}\tan\frac{\omega T}{2}) \qquad (2.2.17)$$

Taking the negative of the derivative of equation(2.2.17), we have

$$\phi(\omega) = \sec^2(\frac{\omega T}{2})\tau(\frac{2}{T}\tan\frac{\omega T}{2}) \qquad (2.2.18)$$

or

$$\phi(\omega) = (1+\frac{T^2}{4}\Omega^2)\tau(\Omega) \qquad (2.2.19)$$

where Ω is given by equation(2.2.11), as $\Omega = \frac{2}{T}\tan(\frac{\omega T}{2})$. Thus $\phi(\omega)$ is a distorted version of $\tau(\Omega)$, except for small values of ω, where the Bilinear Z Transform is almost linear.

2.2.3 THE MATCHED Z TRANSFORM:

This is a s-plane to z-plane mapping of the poles and zeros of the analogue filter by the substitution

$$(s-s_i) \Longrightarrow (1- e^{s_iT}.z^{-1}) \qquad (2.2.20)$$

This transform ensures that the poles of the digital filter H(z) will be identical to those obtained by the Impulse Invariance Transform, however, the zeros will not correspond.

Although this transform is easy to apply, there are many cases which it is not suitable. If the analogue filter has zeros with centre frequencies greater than half the sampling frequency, their z-plane positions will be greatly aliased, hence distortion results. Another case happens when the analogue transfer function is an all-pole function. Then the derived digital filter will be an all-pole system such that in many circumstances, does not represent the desired analogue system adequately.

In view of the above, the Impulse Invariance Transformation or the Bilinear Z Transformation is generally preferred over the Matched Z Transform.

2.2.4 FREQUENCY TRANSFORMATIONS: (16),(17),(18),~~(19)~~

A digital filter can also be designed through a transform applied to a suitable lowpass filter via the following schemes:

SCHEME ONE: CONTINUOUS FREQUENCY BAND TRANSFORMATIONS

(a) LOWPASS FILTER ($\Omega_c=1$) TO LOWPASS FILTER ($\Omega_c=\Omega_u$)

$$s \longrightarrow \frac{s}{\Omega_u} \qquad (2.2.21)$$

(b) LOWPASS FILTER ($\Omega_c=1$) TO HIGHPASS FILTER ($\Omega_c=\Omega_u$)

$$s \longrightarrow \frac{\Omega_u}{s} \qquad (2.2.22)$$

(c) LOWPASS FILTER (Ω_c=1) TO BANDPASS FILTER (Ω_L, Ω_u)

$$s \longrightarrow \frac{s^2+\Omega_L.\Omega_u}{s(\Omega_u-\Omega_L)} \qquad (2.2.23)$$

(d) LOWPASS FILTER (Ω_c=1) TO BANDSTOP FILTER (Ω_L, Ω_u)

$$s \longrightarrow \frac{s(\Omega_u-\Omega_L)}{s^2+\Omega_u.\Omega_L} \qquad (2.2.24)$$

where Ω_u, Ω_L are upper and lower cutoff frequencies respectively.

These transformations may result in a higher order filter and ~~is also very non-linear in nature~~ *the frequency axis is non-linearly transformed*. Hence the use of frequency transformations should only be restricted to linear filters with piecewise constant desired characteristics.

SCHEME TWO: DIGITAL FREQUENCY BAND TRANSFORMATIONS

DESIGN ANALOGUE LOWPASS FILTER	→	DIGITIZE FILTER	→	APPLY FREQUENCY BAND TRANSFORM

Ω_c= 1 DIGITAL TO DIGITAL DESIRED FILTER

(a) LOWPASS FILTER ω_c TO LOWPASS FILTER ω_u

$$z^{-1} \longrightarrow \frac{z^{-1}-\alpha}{1-\alpha z^{-1}} \qquad (2.2.25)$$

where $$\alpha = \frac{\sin\left(\left(\frac{1}{2}(\omega_c-\omega_u)\right)T\right)}{\sin\left(\left(\frac{1}{2}(\omega_c+\omega_u)\right)T\right)} \qquad (2.2.26)$$

and ω_u = desired lowpass filter cutoff frequency.

(b) LOWPASS FILTER ω_c TO HIGHPASS FILTER ω_u

$$z^{-1} \longrightarrow \frac{z^{-1}+\alpha}{1+\alpha z^{-1}} \qquad (2.2.27)$$

where $$\alpha = -\frac{\cos\left(\left(\frac{1}{2}(\omega_c-\omega_u)\right)T\right)}{\cos\left(\left(\frac{1}{2}(\omega_c+\omega_u)\right)T\right)} \qquad (2.2.28)$$

and ω_u = desired highpass filter cutoff frequency.

(c) LOWPASS FILTER ω_c TO BANDPASS FILTER ω_o

$$z^{-1} \longrightarrow - \frac{z^{-2} - (2\alpha k/(k+1))z^{-1} + (k-1)/(k+1)}{((k-1)/(k+1))z^{-2} - (2\alpha k/(k+1))z^{-1} + 1} \qquad (2.2.29)$$

where $\alpha = \cos\omega_o T$

$$= \frac{\cos(\left(\frac{1}{2}(\omega_u + \omega_l)\right)T)}{\cos(\left(\frac{1}{2}(\omega_u - \omega_l)\right)T)} \qquad (2.2.30)$$

$$k = \cot(\tfrac{1}{2}(\omega_u - \omega_l)T)\tan(\tfrac{1}{2}\omega_c T) \qquad (2.2.31)$$

and ω_o= desired centre frequency of bandpass filter.

(d) LOWPASS FILTER ω_c TO BANDSTOP FILTER ω_o

$$z^{-1} \longrightarrow \frac{z^{-2} - (2\alpha/(1+k))z^{-1} + (1-k)/(1+k)}{((1-k)/(1+k))z^{-2} - (2\alpha/(1+k))z^{-1} + 1} \qquad (2.2.32)$$

where $\alpha = \cos\omega_o T$

$$= \frac{\cos(\left(\frac{1}{2}(\omega_u - \omega_l)\right)T)}{\cos(\left(\frac{1}{2}(\omega_u + \omega_l)\right)T)} \qquad (2.2.33)$$

$$k = \tan(\tfrac{1}{2}(\omega_u - \omega_l)T)\tan(\tfrac{1}{2}\omega_c T) \qquad (2.2.34)$$

and ω_o= desired centre frequency of reject band of bandstop filter.

These frequency transformations may result in a filter of a higher order. They are of the all-pass type which maps the unit circle into itself one or more times. Thus the frequency scale is warped by the mapping but the magnitude of the lowpass prototype is being preserved. For instance, an elliptic filter will be transformed into new elliptic filters. These digital frequency band transformations are easy to use since they map a rational transfer function into a new rational function. Since they introduce no distortion errors in the amplitude scale, they provide an alternative to the continuous frequency band transformations as a means of obtaining transformed filters from a lowpass prototype. Thus scheme one and scheme two both lead to the same result and

TABLE 2.2.1 gives the transformed frequencies for the digital frequency band transformations.

Critical Frequency Of Transformed Filter	Frequency Of Prototype Filter 0	$-\omega_c$	$+\omega_c$	$\frac{1}{2}\omega_s$
Lowpass	0	$-\omega_u$	$+\omega_u$	$\frac{1}{2}\omega_s$
Highpass	$\frac{1}{2}\omega_s$	$+\omega_u$	$-\omega_u$	0
Bandpass	$\pm\omega_0$	$\pm\omega_l$	$\pm\omega_u$	$0, \frac{1}{2}\omega_s$
Bandstop	$0, \frac{1}{2}\omega_s$	$\pm\omega_u$	$\pm\omega_l$	$\pm\omega_0$

TABLE:2.2.1 TRANSFORMED FREQUENCIES FOR DIGITAL TRANSFORMATIONS

where ω_s= digital filter sampling frequency,

ω_0= centre frequency of bandpass or bandstop filter,

ω_u, ω_l= upper and lower cutoff frequencies.

2.2.5 MAGNITUDE-SQUARED FUNCTION DESIGN: (10)

In the preceding sections we have discussed techniques for digitizing continuous-time analogue filters. The following sections are purely devoted to existing direct designs of digital filters in either the time or frequency domain as opposed to the previous indirect methods. We shall first review the Magnitude-Squared Function design which is a method in the frequency domain whereby we let H(z) to be the Z Transform of an arbitrarily realizble IIR digital filter such that H(z) is of the form

$$H(z) = \frac{\sum_{i=0}^{m-1} b_i . z^{-i}}{\sum_{i=0}^{n-1} a_i . z^{-i}} \qquad (2.2.35)$$

The Magnitude-Squared response of the filter is easily evaluated as

$$|H(e^{j\omega T})|^2 = H(z).H^{-1}(z)\Big|_{z=e^{j\omega T}} \qquad (2.2.36)$$

and can written as

$$|H(e^{j\omega T})|^2 = \frac{\sum_{i=i}^{m-1} b_i . z^{-i}}{\sum_{i=o}^{n-1} a_i . z^{-i}} \cdot \frac{\sum_{k=o}^{m-1} b_k . z^{k}}{\sum_{k=o}^{n-1} a_k . z^{k}} \qquad (2.2.37)$$

$$\text{i.e.} \qquad |H(e^{j\omega T})|^2 = \frac{\sum_{i=o}^{m-1} b_i . e^{-j\omega iT}}{\sum_{i=o}^{n-1} a_i . e^{-j\omega iT}} \cdot \frac{\sum_{k=o}^{m-1} b_k . e^{j\omega kT}}{\sum_{k=o}^{n-1} a_k . e^{j\omega kT}} \qquad (2.2.38)$$

$$\text{or,} \qquad |H(e^{j\omega T})|^2 = \frac{\sum_{i=o}^{m-1} c_i . \cos(\omega iT)}{\sum_{i=o}^{n-1} d_i . \cos(\omega iT)} \qquad (2.2.39)$$

where $\{c_i\}$ and $\{d_i\}$ are related to $\{b_i\}$ and $\{a_i\}$, as can be also written as

$$|H(e^{j\omega T})|^2 = \frac{\sum_{i=o}^{m-1} e_i . \cos^2(\tfrac{1}{2}\omega iT)}{\sum_{i=o}^{n-1} f_i . \cos^2(\tfrac{1}{2}\omega iT)} \qquad (2.2.40)$$

thus showing that the magnitude-squared function can always be expressed as the ratio of two trigonometric functions of ω.

Equation(2.2.40) forms the basis of the Magnitude-Squared Function design which also relates to analogue filters whose magnitude-squared function is a ratio of polynomials in Ω^2. Using the substitution

$$\Omega = \cos(\tfrac{1}{2}\omega T) \qquad (2.2.41)$$

equation(2.2.40) assumes a form appropriate to analogue filter transfer functions, and can also be simplified to the form

$$|H(e^{j\omega T})|^2 = \frac{1}{1 + A_n^2(\omega T)} \qquad (2.2.42)$$

where $A_n^2(\omega T)$ is an nth-order rational trignometric ~~polynomial~~ *function*. By suitable choice of the function $A_n^2(\omega T)$, various types of digital filters can be designed to match the prescribed or desired magnitude characteristics of the transfer functions.

The difficulties with this technique are twofold. First, a suitable rational trigonometric polynomial $A_n^2(\omega T)$ must be found to provide the desired filtering. Secondly, the magnitude-squared function $|H(e^{j\omega T})|^2$ must be factorized to find the poles and zeros. This factorization is generally nontrivial and therefore makes this technique an undesirable filter design method.

2.2.6 TIME DOMAIN DIRECT DESIGN OF IIR DIGITAL FILTERS: (20)-(23)

Since it is possible to design filters to approximate an arbitrary frequency response, it is also possible to design an IIR filter whose impulse response approximates a desired impulse response. Let the Z Transform of the digital filter be of the form

$$H(z) = \frac{\sum_{i=o}^{m-1} b_i . z^{-i}}{\sum_{i=o}^{n-1} a_i . z^{-i}} \qquad (2.2.43)$$

$$= \sum_{k=o}^{\infty} h_k . z^{-k} \qquad (2.2.44)$$

and it is required that the filter impulse response h_k should approximate a desired impulse response g_k, over the range $0 \leqslant k \leqslant p-1$ as closely as possible.

Define an error sequence of the above approximation as the equation

$$e_k = \sum_{k=o}^{p-1} (g_k - h_k)^2 w_k \qquad (2.2.45)$$

where w_k is a positive weighting function on the error sequence.

It has been shown in the above references that under a variety of conditions, a set of $\{a_i, b_i\}$ can be found such that e_k is minimized over all possible choices of a_i and b_i.

Moreover, since h_k is a nonlinear function of the digital filter parameters $(\{a_i\}, \{b_i\})$, generally the minimization of e_k can only be acheeved using iterative techniques. In the case when $p=n+m-1$, the filter parameters to minimize e_k can be obtained by solving $(n+m)$ linear equations. The above procedure involves writing down the filter impulse response directly as,(assuming that $a_o=b_o=1$)

$$g_k = -a_1 \cdot h_{k-1} - a_2 \cdot h_{k-2} - \ldots - a_n \cdot h_{k-n} + b_k \qquad 1 < k \leq m \qquad (2.2.44)$$

$$g_k = -a_1 \cdot h_{k-1} - a_2 \cdot h_{k-2} - \ldots - a_n \cdot h_{k-n} \qquad k > m \qquad (2.2.45)$$

Letting $g_k = h_k$ for k=1, 2, ...,m, equation(2.2.45) can be solved for the set of $\{a_i\}$ that give $g_k = h_k$ for k=m+1, m+2,...,m+n. Given the $\{a_i\}$, equation(2.2.44) can be solved for the $\{b_i\}$ that give $g_k = h_k$, k=1, 2,....,m. This procedure amounts to equating the first (n+m+1) terms in a series expansion of H(z) of equation (2.2.43) to the truncated Z Transform of the desired impulse response g_k after (n+m) terms. This method of approximation of a power series by a rational function is often referred to as the Pade approximant procedure (21). By approximating the desired digital filter response by reproducing the first (n+m+1) samples, it is hoped that the overall time or frequency response of the approximating filter will not deviate too much from the desired one. In practice, there is no simple way of even approximately bounding the deviation on either of these responses, while the iterative techniques involved are best performed on a computer. *In fact, there is no guarantee that the IIR filter so obtained will be stable.*

2.2.7 LINEAR PROGRAMMING DESIGN OF IIR FILTERS: (24),(25)

Just as it is possible to use iterative techniques in the time domain to directly design digital filters, it is also possible to use linear programming technique to design IIR filters in the frequency domain. Equation(2.2.40) in section 2.2.5 can also be written as

$$|H(e^{j\omega T})|^2 = \frac{P(\omega T)}{Q(\omega T)} = \frac{c_o + \sum_{i=1}^{m} c_i \cos(\omega i T)}{d_o + \sum_{i=1}^{n} d_i \cos(\omega i T)} \qquad (2.2.46)$$

where both $P(\omega T)$ and $Q(\omega T)$ are linear in coefficients c_i and d_i. Now linear programming techniques can be used to determine the $\{c_i\}$ and $\{d_i\}$ such that $|H(e^{j\omega T})|$ approximates a given magnitude-squared characteristic $F(\omega T)$ where the peak approximation error is minimized i.e. an equiripple approximation.

If we let $F(\omega T)$ be the desired magnitude-squared characteristic then the approximation problem consists of finding the filter

coefficients such that

$$-e(\omega T) \leqslant \frac{P(\omega T)}{Q(\omega T)} - F(\omega T) \leqslant e(\omega T) \qquad (2.2.47)$$

where $e(\omega T)$ is a tolerance function on the error that allows for unequal weighting of errors as a function of frequency. Since $F(\omega T)$ and $e(\omega T)$ are generally known functions of frequency, equation (2.2.47) can be expressed as a set of linear inequalities in the values of $\{c_i\}$ and $\{d_i\}$ by writing it in the form

$$\begin{aligned} P(\omega T) - Q(\omega T).F(\omega T) &\leqslant e(\omega T).Q(\omega T) \\ -P(\omega T) + Q(\omega T).F(\omega T) &\leqslant e(\omega T).Q(\omega T) \end{aligned} \qquad (2.2.48)$$

or

$$\begin{aligned} P(\omega T) - Q(\omega T)(F(\omega T) + e(\omega T)) &\leqslant 0 \\ -P(\omega T) + Q(\omega T)(F(\omega T) - e(\omega T)) &\leqslant 0 \end{aligned} \qquad (2.2.49)$$

with the additional linear inequalities

$$-P(\omega T) \leqslant 0 \qquad (2.2.50)$$

$$-Q(\omega T) \leqslant 0 \qquad (2.2.51)$$

which together completely define the approximation problem as an exercise in linear programming. In general, however, this method is subject to accuracy problems related to coefficient sensitivity in magnitude-squared function design described previously.

2.2.8 OPTIMIZATION METHODS FOR DESIGNING IIR FILTERS: (26)

Assume the IIR filter's Z Transform is of the form

$$H(z) = A\prod_{k=1}^{K} \frac{1 + a_k.z^{-1} + b_k.z^{-2}}{1 + c_k.z^{-1} + d_k.z^{-2}} \qquad (2.2.52)$$

so that the filter is realized as a cascade of second-order sections. Let the desired magnitude response of the filter be denoted as $|H_d(e^{j\omega T})|$. Let $\{\omega_i,\ i=1,\ 2,\ldots,M\}$ be the discrete set of frequencies (not necessary to be uniformly spaced) at which the error between the actual response and desired response is evaluated. Then the squared error in frequency (as a function of the filter parameters) can be expressed as

$$e(\theta) = \sum_{i=1}^{M} \left(|H(e^{j\omega_i T})| - |H_d(e^{j\omega_i T})| \right)^2 \qquad (2.2.53)$$

where θ is the vector, with 4K+1 components, of unknown coefficients

$$\theta = (a_1, b_1, c_1, d_1, \ldots, a_K, b_K, c_K, d_K, A) \qquad (2.2.54)$$

To minimize the squared error in equation(2.2.53), implies finding the optimum value of θ, say θ', such that

$$e(\theta') \leqslant e(\theta) \qquad \text{for } \theta \neq \theta' \qquad (2.2.55)$$

The optimization problem involved requires the use of a nonlinear optimization procedure such as the Fletcher-Powell algorithm (26).

2.2.9 WINDOWING TECHNIQUE: (10),(11),(14),(27)

Having dealt with existing approaches,both direct and indirect, to design IIR digital filters, we now turn to a class of digital filters known as the Finite Impulse Response (FIR) digital filters.

Since $H(e^{j\omega T})$, the frequency response of any digital filter, is periodic in frequency, it can be expanded in a Fourier Series. The resultant series is then of the form

$$H(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} h_n . e^{-j\omega nT} \qquad (2.2.56)$$

where

$$h_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega T}) . e^{j\omega nT} d\omega T \qquad (2.2.57)$$

The coefficients of the Fourier series, h_n, are easily recognized as being identical to the impulse response of a digital filter.

One possible way of obtaining an FIR filter that approximates $H(e^{j\omega T})$, would be to truncate the infinite Fourier series ;i.e. to truncate the infinite impulse response to obtain a finite-duration impulse response,h_n^T, such that,

$$h_n^T = \begin{cases} h_n & 0 \leqslant n \leqslant N-1 \\ 0 & \text{otherwise} \end{cases} \qquad (2.2.58)$$

However, direct truncation of the series leads to the well-known Gibbs phenomenon, which manifests itself as a fixed percentage overshoot and ripple before and after an approximated discontinuity in the frequency response. Therefore, in designing good or close approximations to ideal filters, direct truncation of the infinite impulse response is not a reasonable way of obtaining an FIR filter.

A more successful way of obtaining an FIR filter is to use a finite weighting sequence w_n, called a Window, to modify the Fourier coefficients h_n in equation(2.2.56) to control the convergence of the Fourier series. In general, we can represent a finite-duration impulse response g_n as the product of the ideal or desired impulse response h_n and a finite-duration Window, that is,

$$g_n = h_n . w_n \qquad (2.2.59)$$

Using the complex convolution theorem, it can be shown that

$$G(e^{j\omega T}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\theta}) . W(e^{j(\omega T-\theta)}) d\theta \qquad (2.2.60)$$

That is, $G(e^{j\omega T})$ is the periodic convolution of the desired or ideal frequency response $H(e^{j\omega T})$ with the Fourier Transform of the Window function $W(e^{j\omega T})$. The technique of windowing is illustrated in fig. 2.2.6. In fig. 2.2.6a is shown a specification of an ideal filter. Since the impulse response is of length N, then only N samples H_n of the continuous frequency response can be specified as shown. Fig. 2.2.6b shows the actual frequency response which exhibits strong sidelobes due to a direct truncation of the infinite series in equation(2.2.56). Fig. 2.2.6c and 2.2.6d show a window function $W(e^{j\omega T})$ and its samples W_n. When W_n is convolved with H_n, we get G_n as shown in fig. 2.2.6e. The resultant frequency spectrum $G(e^{j\omega T})$ has sidelobes much reduced but the transition band widened. The in-band ripples and overshoots are also suppressed in the windowed frequency response as shown in fig. 2.2.6f.

From the example above, it is seen that there are several noteworthy effects of windowing the Fourier coefficients of the ideal filter on the resulting FIR filter response. A major effect is that discontinuities in $H(e^{j\omega T})$ become transition bands, and in-band and out-of-band ripples are reduced at the expense of a widened transiton

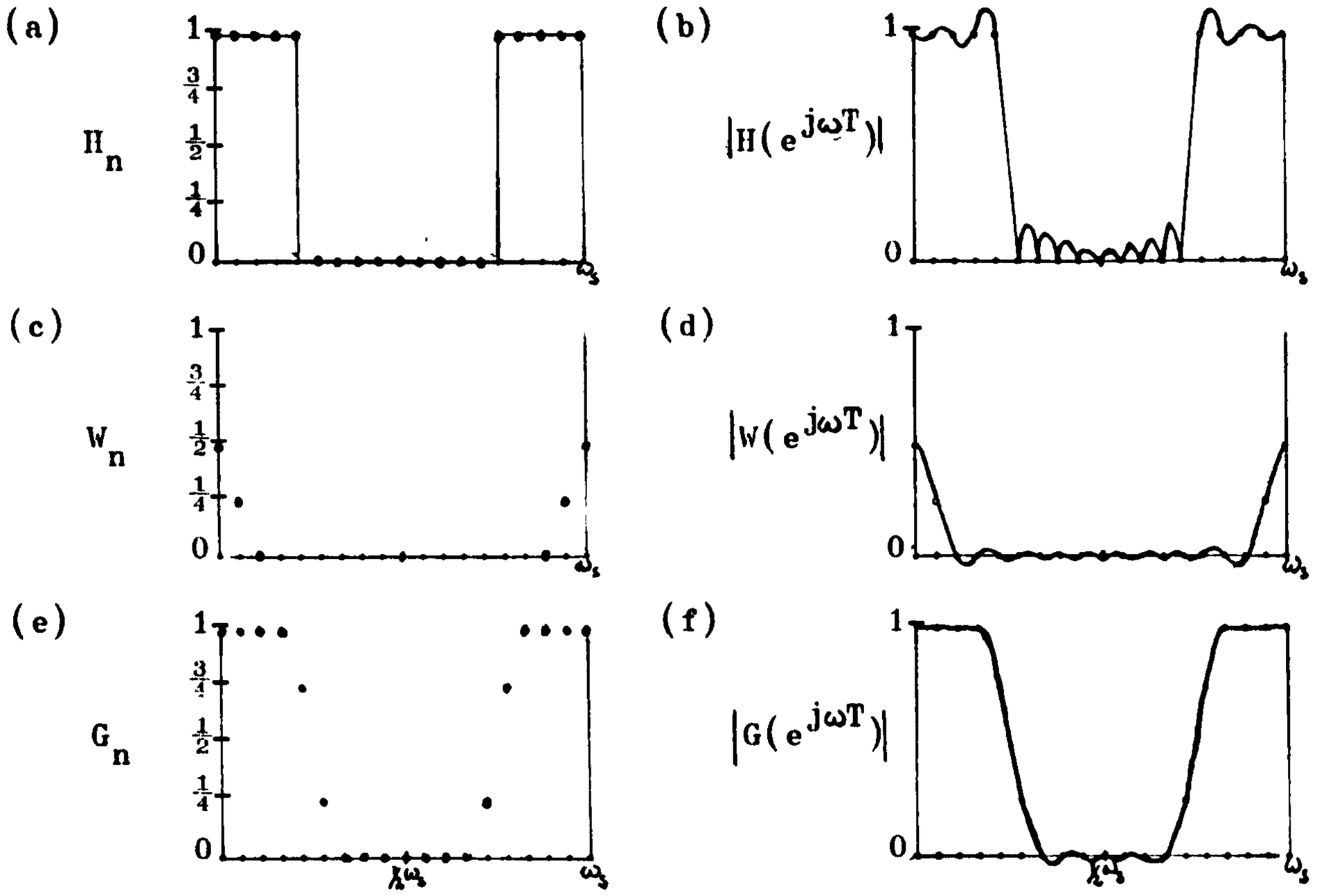


FIG: 2.2.6 WINDOWING TECHNIQUE

band which implies less sharp frequency cutoff. Since the final frequency response of the filter is the ~~circular~~ convolution of the ideal frequency response and the Window's frequency response, it is clear that the width of these transition bands depends on the width of the main lobe of $W(e^{j\omega T})$. Thus, in practice, one must compromise in the choice of windows for minimizing the out-of-band sidelobes and in-band ripples. A secondary effect of windowing is that the ripple from the sidelobes of $W(e^{j\omega T})$ produces approximation errors (ripple in the resulting frequency response) for all . This effect has to be suppressed by tapering the sidelobes of any window used. Finally, since the filter frequency response is obtained via a convolution relationship, it is clear that the resulting filters are never "optimal" in any sense, even though the windows from which they are obtained may satisfy some reasonable optimality criterion.

Ideally, window characteristics are:

(a) Small width of main lobe of the frequency response of the window containing as much of the total energy as possible.

(b) Sidelobes of the frequency response that decrease in energy

rapidly as ω tends to $\frac{\omega_s}{2}$.

There have been many windows proposed that approximate the desired characteristics. They include: the Rectangular Window, the Bartlett Window, the Hanning Window, the Hamming Window, the Blackman Window, the Kaiser Window and the Dolph-Chebyshev Window. Their comparisons and equations are given in the references quoted above, and will not be repeated here.

2.2.10 FREQUENCY SAMPLING DESIGN: (28)-(36)

A FIR filter can be uniquely specified by giving either the impulse response coefficients h_n or equivalently the Discrete Fourier Transform (DFT) coefficients H_k, where

$$H_k = \sum_{n=o}^{N-1} h_n . e^{-j(\frac{2\pi}{N})nk} \qquad \text{DFT} \qquad (2.2.61)$$

and

$$h_n = \frac{1}{N} \sum_{k=o}^{N-1} H_k . e^{j(\frac{2\pi}{N})nk} \qquad \text{IDFT} \qquad (2.2.62)$$

It is further noticed that the DFT samples for a FIR sequence can be regarded as samples of the digital filter's Z Transform evaluated at N points equally spaced around the unit circle, that is

$$H_k = H(z)\Big|_{z=e^{j(\frac{2\pi}{N})k}} \qquad (2.2.63)$$

Thus the Z Transform of a FIR filter can easily be expressed in terms of its DFT coefficients by putting

$$H(z) = \sum_{n=o}^{N-1} h_n . z^{-n}$$

$$= \sum_{n=o}^{N-1} \left(\frac{1}{N} \sum_{k=o}^{N-1} H_k . e^{j(\frac{2\pi}{N})nk} \right) z^{-n} \qquad (2.2.64)$$

Interchanging orders of summation and summing over the index n, gives

$$H(z) = \sum_{k=o}^{N-1} \frac{H_k}{N} \left[\sum_{n=o}^{N-1} (e^{j(\frac{2\pi}{N})k} . z^{-1})^n \right]$$

$$= \sum_{k=o}^{N-1} \frac{H_k}{N} \left[\frac{1 - e^{j2\pi k} . z^{-N}}{1 - e^{j(\frac{2\pi}{N})k} . z^{-1}} \right] \qquad (2.2.65)$$

Since $e^{j2\pi k}=1$, therefore,

$$H(z) = \frac{1-z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H_k}{1-z^{-1}.e^{j(\frac{2\pi}{N})k}} \qquad (2.2.66)$$

Hence, this shows that to approximate any continuous frequency response, one would sample in frequency at N equi-spaced points, called the frequency samples, around the unit circle, and evaluate the continuous frequency response as an interpolation of the sampled frequency response. The approximation error will then be exactly zero at the sampling points and finite between them. The smoother the frequency response being approximated, the smaller the error of the interpolation between the samples, and hence, in general, N has to be large for a satisfactory result.

Furthermore, filters designed by the above method have exactly linear phase in the passband. A sharp transition from the passband to the stopband with a considerable amount of attenuation is often desired. Hence, the frequency samples in the transition band should be weighted as in equation (2.2.66), or optimized to give the desired response. The structure of the frequency sampling design is depicted in fig. 2.2.7. It consists of a comb filter cascaded with N first-order digital resonators tuned at the various sampling points.

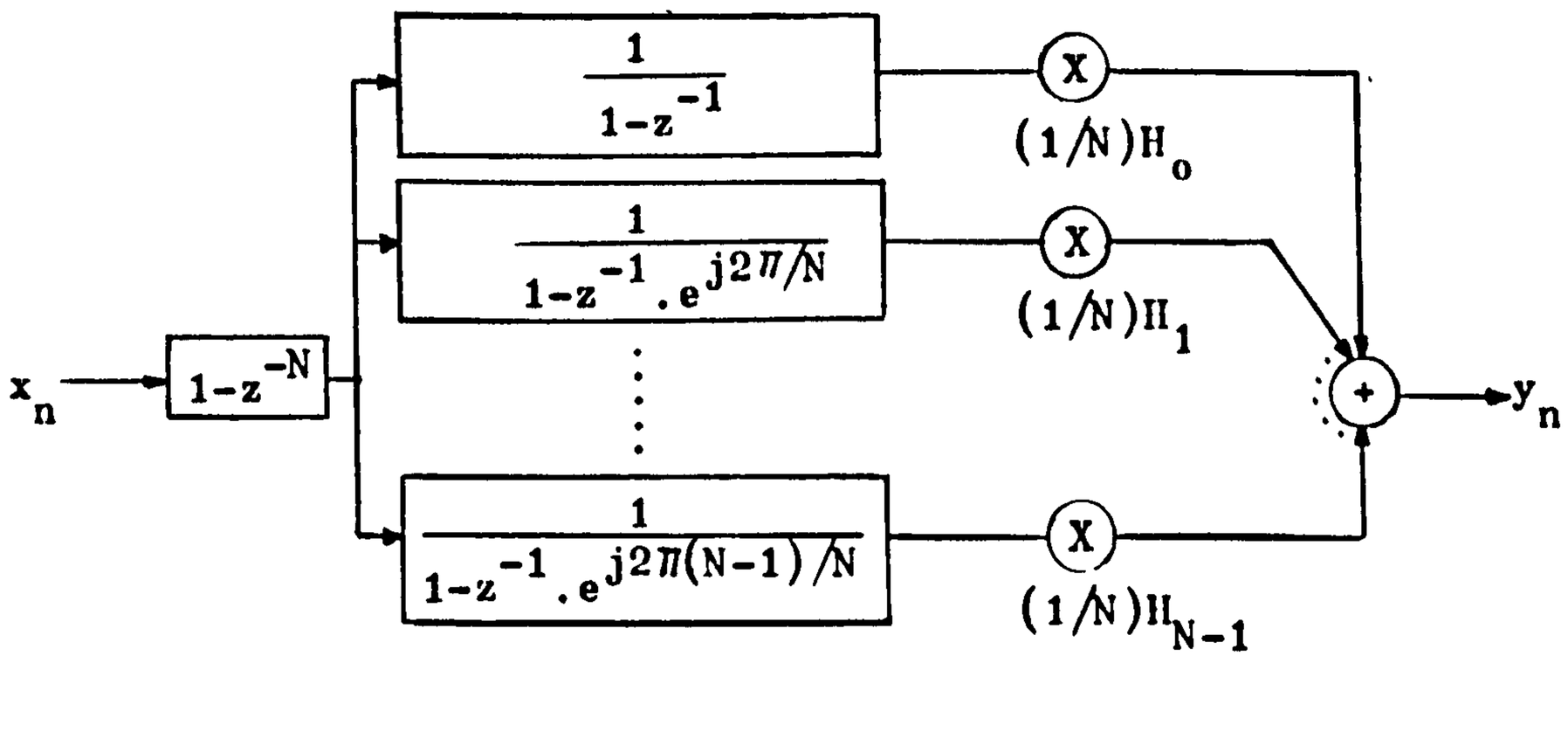


FIG: 2.2.7 FREQUENCY SAMPLING STRUCTURE

This frequency sampling design has several interesting properties. It can be regarded as a pole-zero cancelling

combination. The comb filter provides N zeros which are selectively cancelled by the following N resonators which provide the required poles in the cancellation. However, when using finite precision arithmetic to represent the frequency sampling structure, the poles of the filter will not cancel the zeros of the comb filter exactly. Hence due to quantization effects, it is generally advised to move the zeros and the poles of the frequency sampling design slightly inside the unit circle, with a radius of something like $R=1-2^{-27}$ for complete cancellation to occur, otherwise, the design will have both poles and zeros remained uncancelled (although exact cancellation of poles and zeros using finite precision arithmetic is difficult, in practice, any design with errors within a prescribed tolerance scheme will be deemed to exhibit complete cancellation). This is especially true when small wordlengths are used to program the filter. Thus, the design imposes stringent constraints on the choice of wordlength used. The fact that longer wordlengths have to be used may offset the attractiveness of the frequency sampling design when one comes to implement the digital filter with a special-purpose processor. Another snag is the volume of hardware required to implement the comb filter, particularly with large N.

The importance of the design lies in the fact that it is an extremely efficient realization of a filter for which the majority of the filter coefficients H_k are exactly zero and for those paths the resonators do not have to be realized at all. Hence, frequency sampling designs are particularly suitable for narrowband frequency selective filters where only a few of the samples of the frequency response are non-zero. In such cases, a frequency sampling design may be considerably more efficient than either direct convolution or convolution via the DFT. In general, even if more than a few samples are non-zero, the frequency sampling design yields excellent results. However, this method lacks flexibility in specifying the passband and stopband cutoff frequencies since the placement of the frequency samples is constrained to integer multiples of $2\pi/N$. By making N sufficiently large, samples can be obtained arbitrarily close to any given frequency. Therefore, relative flexibility in specifying frequency samples can be achieved at the expense of large N.

2.2.11 GENERAL COMMENTS AND CONCLUSIONS:

We have presented a number of design techniques for IIR and FIR digital filters. As is generally the case, it is impossible to say that one specific filter design is recommended to the exclusion of the others. Consequently, depending on the specific application in mind, the availability of the various design techniques, and numerous otherfactors, almost any of the design techniques described previously may be most preferred.

In general, the choice between an FIR filter and an IIR will depend on the relative weight that one attaches to the advantages and disadvantages of each type of filter. IIR filters, for example, have the advantage that a variety of frequency selective filters can be designed using closed-form design formulas. In the case of FIR filters, closed-form expressions do not exist. Although the window method can be applied in a rather straightforward manner, some iteration may be necessary to meet a prescribed specification. As opposed to IIR filters, most of the other FIR design methods are iterative procedures, requiring rather powerful computational facilities for their implementation.

In general, IIR designs lack flexibility in matching an arbitrary frequency response and are primarily limited to lowpass, bandpass, bandstop and highpass filters, etc. Furthermore, these designs generally disregard the phase response of the filter so that in linear phase applications, a phase equalizer is required. In contrast, FIR filters can have precisely linear phase. Also, the window method and the frequency sampling design afford the possible approximation of rather arbitrary frequency response characteristics with relative ease. Also, it appears that the design problem for FIR filters is much more under control than IIR filters when techniques of optimization are applied.

Finally, there are factors such as cost of implementation and hardware complexity or computational speed of a digital filter. These factors are more or less related to the order of the filter required to meet a given specification. If phase considerations can be neglected, it is generally true that a given amplitude response will be met most efficiently with an IIR filter. However, in some cases, the linear phase available with an FIR filter may well worth the extra cost.

To conclude this section, it is worth pointing that although a large number of design techniques have been discussed and indications to their choice given, there are additional FIR filter designs which can be found in references (10) and (11). It is suffice to say here that these design procedures rely on the techniques of optimization and linear programming such as those mentioned in sections(2.2.7) and (2.2.8). By considering the linear phase FIR filter design problem as a Chebyshev approximation problem it is possible to derive a set of conditions for which it can be proved that the solution is optimal(in the sense that the peak approximation error over the entire interval of approximation is minimized) and unique. This is known as the Weighted Chebyshev Approximation. It is also easy to show how several standard optimization procedures(10) including linear programming can be used to solve for the filter coefficients of the optimal(minimax) solution. These optimization techniques usually result in maximal ripple FIR filters in the sense that they have the maximum number of ripples in their approximations to some desired responses. Polynomial interpolation solution has also been introduced for the above Maximal Ripple FIR filters, which is basically an iterative algorithmic approach for producing a polynomial that has extrema of desired values. In addition, the well-known Remez Exchange Algorithm (10) has also been used to design optimal FIR filters.

CHAPTER THREE

A NOVEL DESIGN TECHNIQUE OF DIGITAL FILTERS FOR HIGH-SPEED REALTIME FILTERING

3.1 INTRODUCTION:

In general, the design of digital filters involves three basic steps. The first step requires the specifications of the desired filters whose properties are highly dependent on their applications. The second step is the approximation of these specifications, using an appropriate design technique, while the final step is the implementation of the filters using finite-precision arithmetic. In the present chapter, we shall propose a design technique which is particularly suitable for high-speed filtering involving a filter bank or a number of channels.

A realtime process is one for which, on the average, the computation associated with each sampling interval can be completed in a time less than or equal to the sampling interval. Consequently, a criterion often imposed on the design of realtime digital filters is the requirement of a minimum amount of computational effort in producing an output sample. While the above situation might not be so critical for low-speed applications, it is certainly crucial for high-speed applications such as radar signal processing where a filter bank is often used to filter the doppler returns. The filter bank normally consists of a large number of narrowband bandpass filters with equal passband characteristics, centred at the possible doppler returns. Thus, in this instance and many other similar circumstances, a great deal of computation has to be performed in realtime.

The design techniques of a number of IIR and FIR digital filters were reviewed in chapter two previously. However, none of them has made substantial contributions to high-speed realtime digital filtering. Their implementations have been mainly in the form of computer software in low-speed applications. While performance is concerned, substantial out-of-band rejection coupled with sharp transition characteristics have been reported for elliptic filters in the case of IIR digital filters, as well as frequency sampling

filters in the case of FIR digital filters. In this instance, high order filters are used in both cases which, on the other hand, necessitate a large amount of computational effort. In addition, the required hardware increases correspondingly, making these filters less economically viable if they are to be implemented in special-purpose realtime processors; let alone the processing speed. Hence, for instance, in the case of a filter bank implementation, an efficient design technique often compromises between performance and operational speed in realtime applications. Furthermore, IIR digital filter designs often disregard the phase response of the resultant filters. Consequently, although we might obtain an elliptic digital filter with excellent amplitude response characteristic, its phase response will be highly nonlinear, especially at the bandedge(s). In contrast, FIR digital filters can have precisely linear phase, but they normally require rather powerful computational facilities for their implementation.

Nevertheless, of all the design techniques reviewed in the previous chapter, the frequency sampling design has a pole-sharing property whereby poles of adjacent filters in a filter bank design can be shared; therefore indirectly reducing the amount of computation involved. In addition, the frequency sampling design yields a high degree of phase linearity, and we shall subsequently examine its "defects" in high-speed realtime filtering thereby leading to the development of our proposed design technique. The proposed technique also has an inherent pole-sharing property which, together with its zero-sharing property, greatly reduces the computational effort to yield an output sample. Realtime calculation of filter coefficients are also feasible via some interpolation formulae and the resultant modularity of the design. This implies the dispencing with the need of coefficients storage, which, in the case of a filter bank, can be quite excessive. Moreover, the above possibility of realtime calculation of coefficients further enhances the efficiency of the proposed design technique of digital filters for high-speed filtering.

The frequency sampling design introduces the sampling of an arbitrary frequency response $H(e^{j\omega T})$ in the frequency domain, thereby providing a direct means of designing digital filters by combining

the individual responses of a set of elemental filters at N equally spaced sampling points. The structure of the frequency sampling design consists of a comb filter in cascade with a number of resonators in parallel. However, owing to the inherent snags of the design, it is not suitable for high-speed digital filtering in realtime using dedicated hardware. The major drawbacks of the frequency sampling design can be briefly recapitulated as follows:

(a) The pole-zero cancellation which necessitates much increased accuracy in the arithmetic used to implement the design. This often entails long wordlengths in most circumstances, thereby increasing both the complexity and the volume of hardware used, as well as decreasing the throughput rate of the processor.

(b) The large amount of memory used in the comb filter section for large N, owing to the fact that the frequency sampling filters are basically FIR digital filters.

(c) The lack of flexibility in specifying the passband and the stopband cutoff frequencies, since the placement of the frequency samples H_k is constrained to integer multiples of $2\pi/N$.

The frequency, phase and time responses of an elemental filter of the frequency sampling design are shown in fig. 3.1.1, where Δt is the finite duration of the elemental filter impulse response and $\Delta\omega$ is the width of the mainlobe of the elemental filter frequency response, such that

$$\Delta t \approx \frac{1}{\Delta f} = \frac{2\pi}{\Delta\omega} \qquad (3.1.1)$$

Here the symbol "$\approx$" has the meaning of " is proportional to and is of the order of ....".

In view of the above, one tends to seek an alternative approach to sample an arbitrary frequency response, without the inherent snags of the frequency sampling design. In our proposed technique, we call for a set of high-Q resonators as elemental filters so that the pole-zero cancellation in the frequency sampling design is avoided. As a result, a much weaker constraint is now imposed on the accuracy of the finite-precision arithmetic used in the hardware implementation. In the following discussion of the proposed technique, simple second-order resonators are used(for a detail analysis of these resonators, the reader is now referred to appendix B). The frequency, phase and

time responses of these second-order elemental filters are shown in fig.3.1.2.

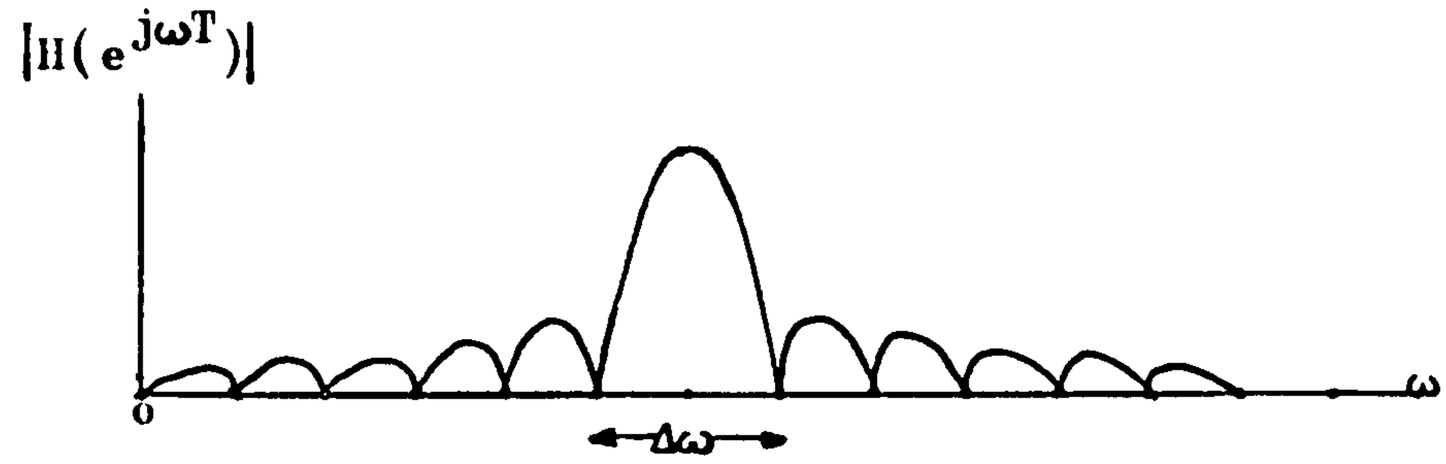


FIG: 3.1.1a FREQUENCY RESPONSE OF AN ELEMENTAL FILTER IN THE FREQUENCY SAMPLING DESIGN

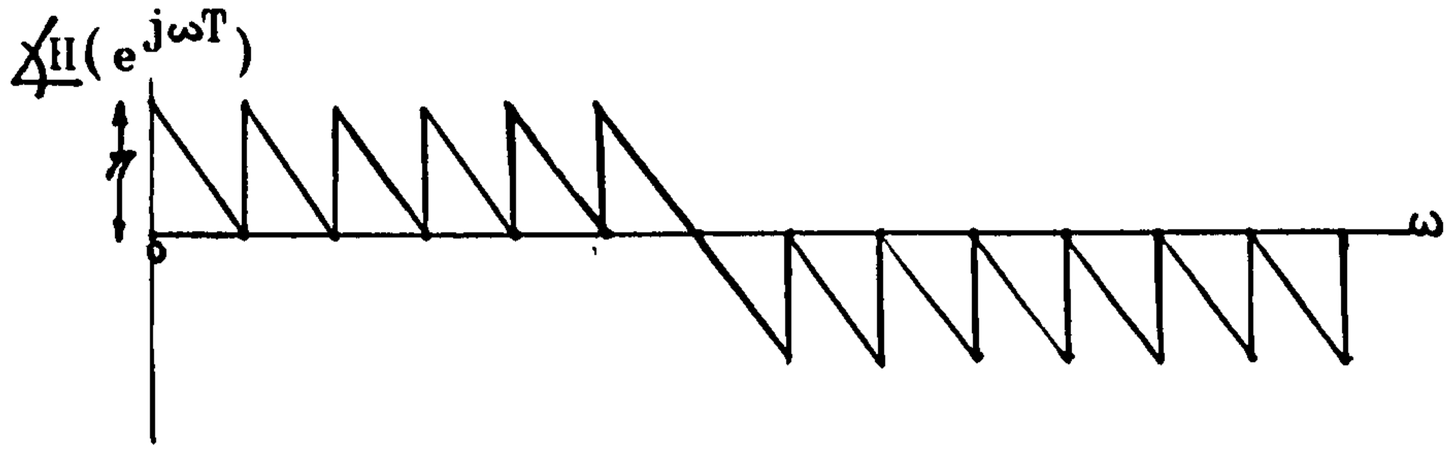


FIG: 3.1.1b PHASE RESPONSE OF AN ELEMENTAL FILTER IN THE FREQUENCY SAMPLING DESIGN

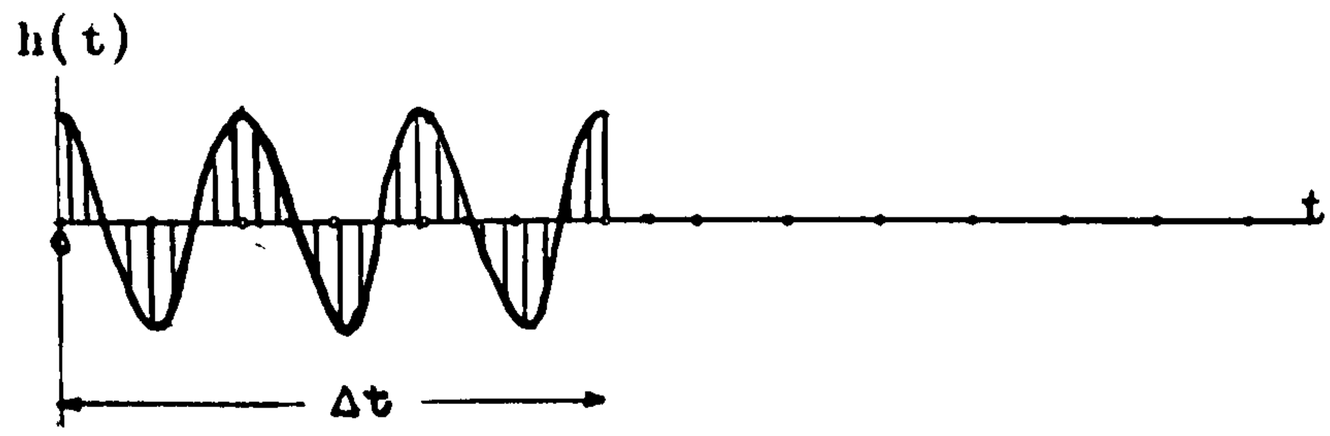


FIG: 3.1.1c TIME RESPONSE OF AN ELEMENTAL FILTER IN THE FREQUENCY SAMPLING DESIGN

It should be pointed out that the time responses shown in figures 3.1.1c and 3.1.2c are in fact the continuous-time equivalents of the discrete-time versions. Thus, the discrete-time version h_n in both cases is given by: $h_n = h(t)|_{t=nT}$ as shown by the vertical lines in both figures. In addition, the impulse response of the elemental filter in the case of the frequency sampling design is seen to be an FIR sequence, while that of the proposed technique is seen to be an IIR sequence. Further comparison of the above

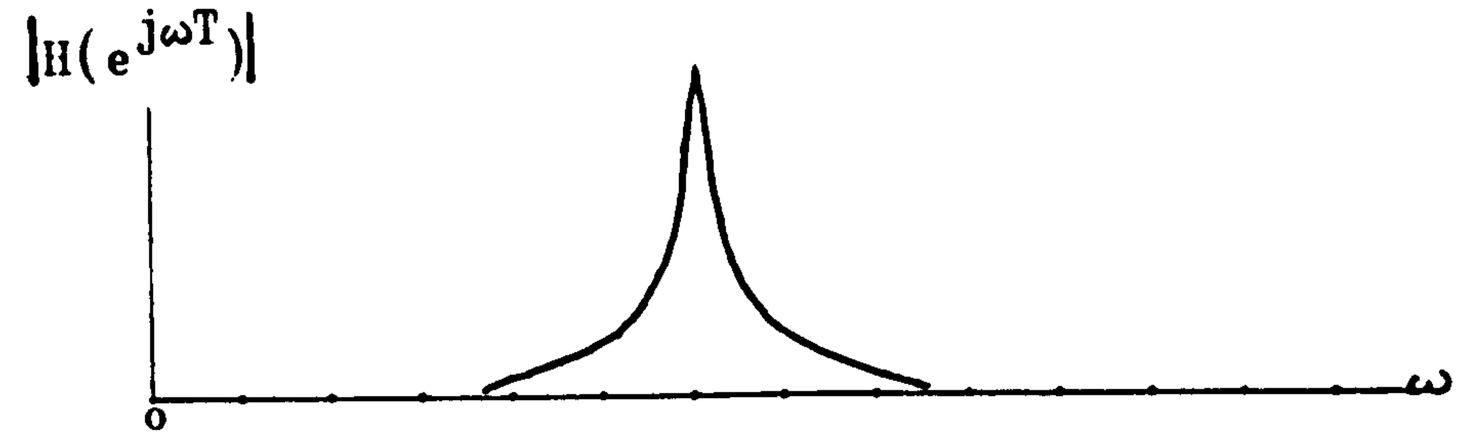


FIG: 3.1.2a FREQUENCY RESPONSE OF AN ELEMENTAL FILTER IN THE PROPOSED DESIGN TECHNIQUE

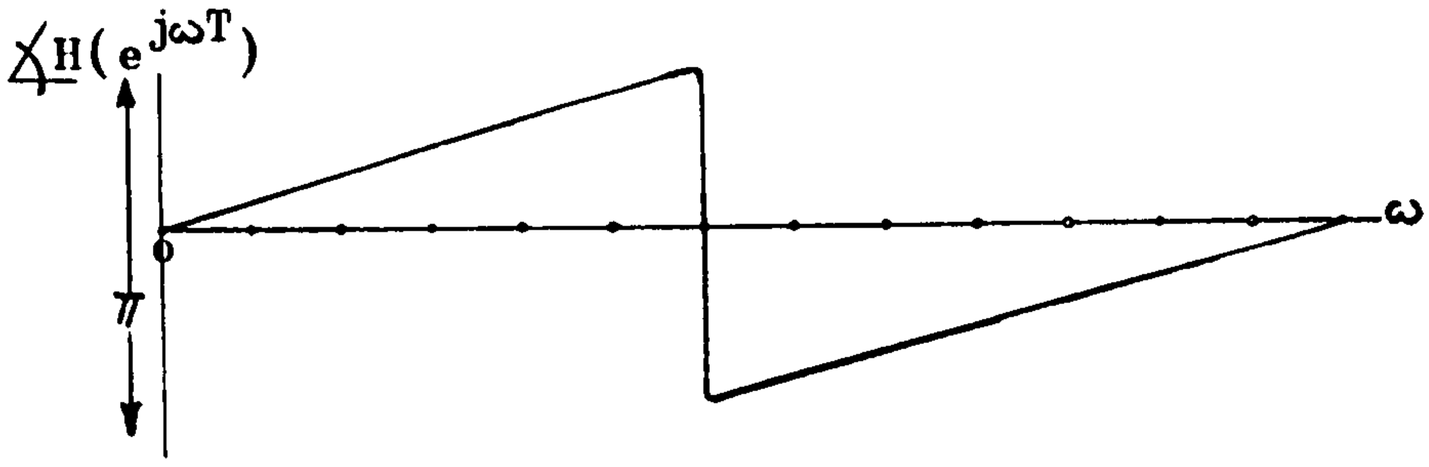


FIG: 3.1.2b PHASE RESPONSE OF AN ELEMENTAL FILTER IN THE PROPOSED DESIGN TECHNIQUE

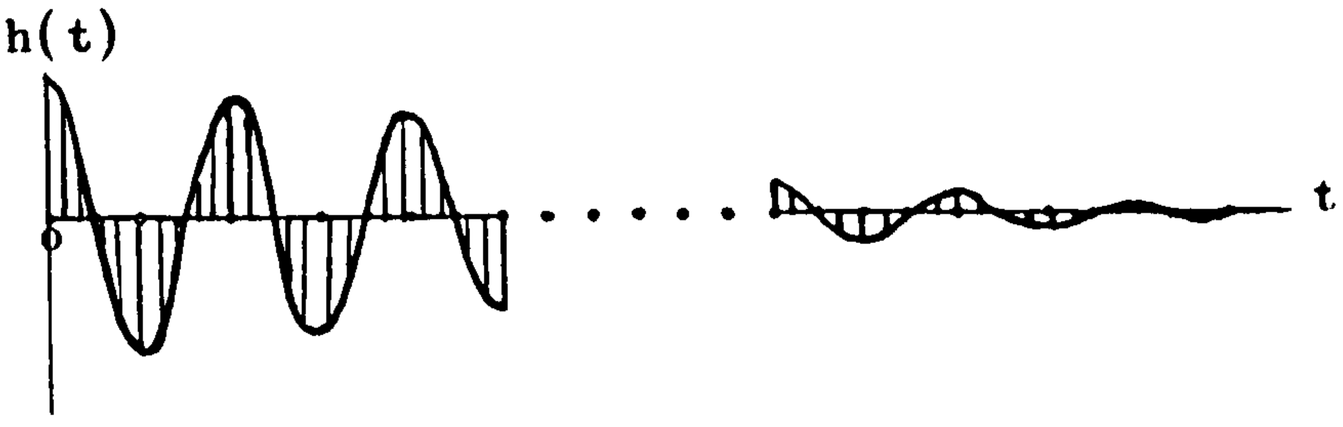


FIG: 3.1.2c TIME RESPONSE OF AN ELEMENTAL FILTER IN THE PROPOSED DESIGN TECHNIQUE

figures shows that the phase response in figure 3.1.1b is exactly linear while that in figure 3.1.2b is only approximate. The frequency response of the elemental filter in the proposed technique does not possess sidelobes at regular intervals as is the case in the frequency sampling design. Nevertheless, both frequency responses exhibit resonance at their resonant frequencies. Thus both designs use resonant elemental filters to sample an arbitrary frequency response to be approximated. Since the elemental filters in the proposed technique do not involve any comb filter, therefore, snag (b) of the frequency sampling design is also evaded. It will be shown later that the spacing of the frequency samples used to

approximate an arbitrary frequency response in the proposed technique is made "flexible", in contrast to the fixed spacing between samples in the frequency sampling design. In particular, frequency samples of the proposed technique are so chosen that their separation is much closer in the transition band(s) than in the passband, thereby providing increased flexibility in locating the cutoff frequencies in the passband and the stopband(s) of the desired response. Consequently, snag(c) of the frequency sampling design has also been overcome, since the placement of the transition frequency samples is no longer restricted in the proposed technique.

Since the elemental filters of the frequency sampling design are orthogonal, the design is mathematically "trivial", though the actual implementation is difficult (its impulse response is on the verge of instability, and finite wordlength effects may push it one way or another). In the proposed technique, resonators with poles slightly within the unit circle are used for stability reason, to provide an easier implementation, however, the elemental filters of the design are no longer orthogonal as opposed to those of the frequency sampling design. Furthermore, the proposed design is complicated by the interactions among elemental filters and an optimization procedure is incorperated in the design to provide resultant filters which are optimum in terms of steep transitions. In addition, the optimization procedure is capable of providing local modifications of the actual response in the approximation to suit the specifications of the desired response. A closed-form expression of the passband ripple is also derived in a later section to facilitate the design and to alleviate the optimization problem. Phase modifications have also been included in the design to provide reasonable phase linearity throughout the passband region. The poles of the resultant design have been located on a circle with a radius slightly less than unity, such that the design exhibits a pole-sharing property in filter bank implementations.

3.2 LINEAR PHASE FREQUENCY RESPONSE APPROXIMATION TECHNIQUE USING OPTIMAL ELEMENTAL FILTERS:

Having briefly introduced our proposed technique for realtime filtering, we now proceed to described it in

details in this section. This approximation technique can be split into two steps. First, we sample in frequency, the ideal spectrum of a digital filter to be approximated. The sampling is done by distributing elemental filter responses, called the frequency samples, uniformly along the passband, so that the entire spectrum of interest is sampled "adequately". Next, we erect frequency samples in the transition band(s) at closer equispaced intervals to improve the transition characteristics. In the second step of the approximation, we optimize the design to provide an optimum structure. In particular, the transition frequency samples are being optimized to yield the steepest transition characteristics possible. Under this optimization procedure, the resultant filter response is tailored to suit individual specifications given, and in general, larger stopband attenuation and much reduced sidelobes can be obtained. In addition, linear phase is approximated in the passband by having all poles at equal distance from the unit circle and also "phase-modified".

3.2.1 THE INFINITE STRUCTURE:

Consider figure 3.2.1 which shows an infinite pole-pattern in the complex z-plane. In this case, the whole of the z-plane bounded by the unit circle, is subdivided into an infinite number of concentric circles with radii R, where R is a number less than unity. The poles on these concentric circles are spaced at equal intervals from each other.

We further observe that this infinite pole-pattern is in essence a grid structure formed by superimposing radial axes on the concentric circles where each grid point is a possible pole location for approximating an arbitrary transfer function H(z). In our approximation approach, poles on circles very close to the unit circle are of particular interest, since they exhibit high-Q or dominant-pole effect. Theoretically, one can locate poles at any radius in the infinite grid structure, however, in actual hardware implementations using finite-precision arithmetic, the number of allowable grid points become finite. If the grid points are "quantized", then one gets a quantized approximation of H(z) as a result. The effect of quantization due to finite word lengths (a detail discussion of finite word length effects is presented in

chapter six) is to move the poles from their otherwise unquantized positions towards the unit circle, and sometimes, even outside it such that instability results. Nevertheless, one can, on the other hand, start off from locating poles at suitably chosen quantized positions so as to finish with a stable quantized design.

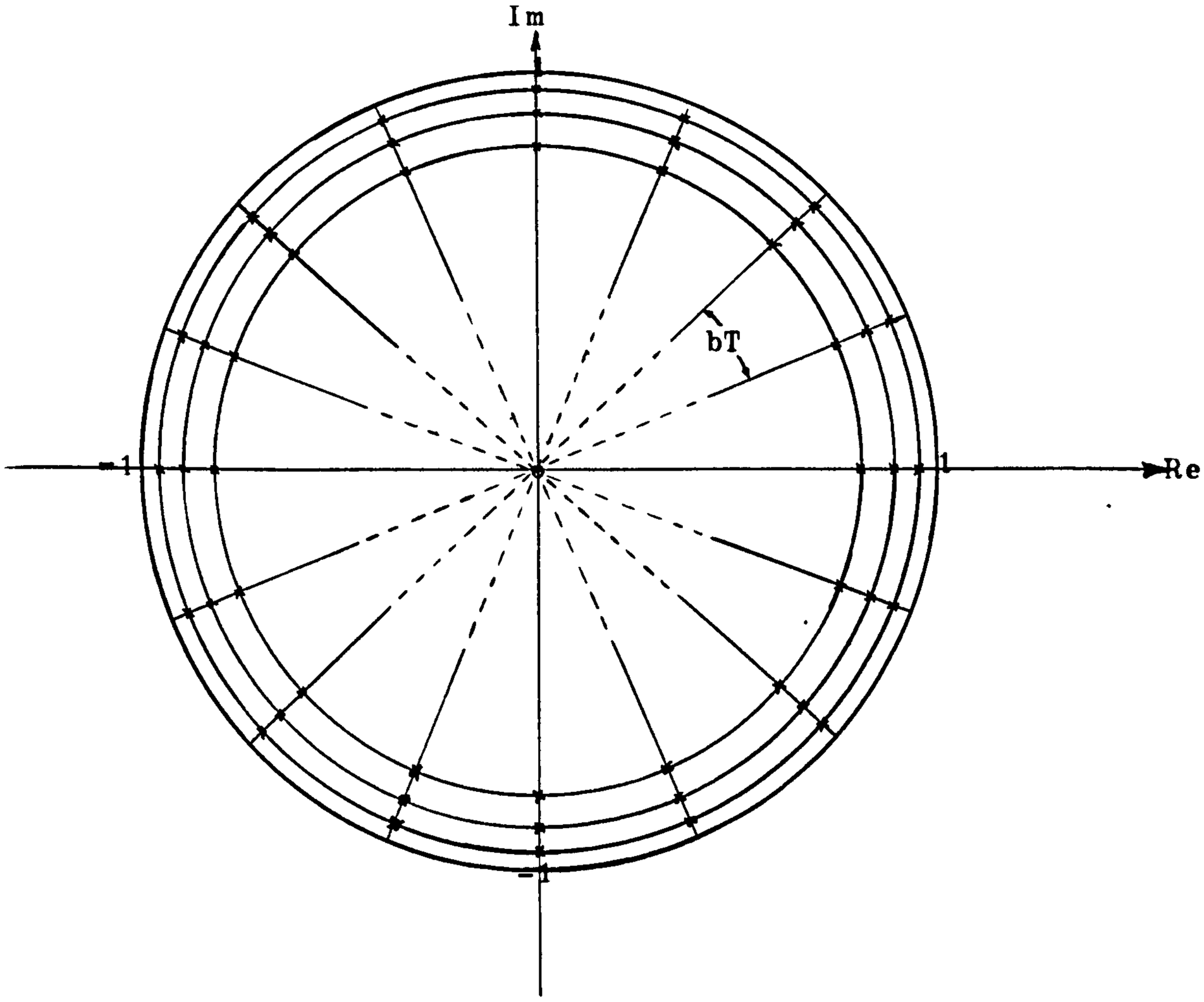


FIG 3.2.1 INFINITE POLE-PATTERN IN THE Z-PLANE

The angle between the radial axes is bT as shown in figure 3.2.1 above, where b is the separation between poles in frequency. The infinite pole-pattern in figure 3.2.1 has an equivalent grid structure in the s-plane as shown in figure 3.2.2. The distance of an arbitrary grid point from the jΩ-axis in the s-plane is a, where $e^{-aT} = R$, and T is the underlying sampling period in the digital system. All poles are equispaced by a distance b in frequency along axes parallel to the jΩ-axis.

The equation of poles on an arbitrary axis parallel to the jΩ-axis can be written as:

$$H_a(s) = \sum_{n=-\infty}^{\infty} \frac{a}{s+(a-jnb)} \qquad (3.2.1)$$

The transfer function for a single pole on the axis is

$$H_p(s) = \frac{a}{s+(a+jnb)} \qquad (3.2.2)$$

while its conjugate pole can be written as

$$H_p^*(s) = \frac{a}{s+(a-jnb)} \qquad (3.2.3)$$

where n varies from positive infinity to negative infinity in integer values in both cases. Hence, we can write the transfer function of the complex pole-pair as

$$H_p'(S) = \frac{(s+a)}{(s+a)^2 + (nb)^2} \qquad (3.2.4)$$

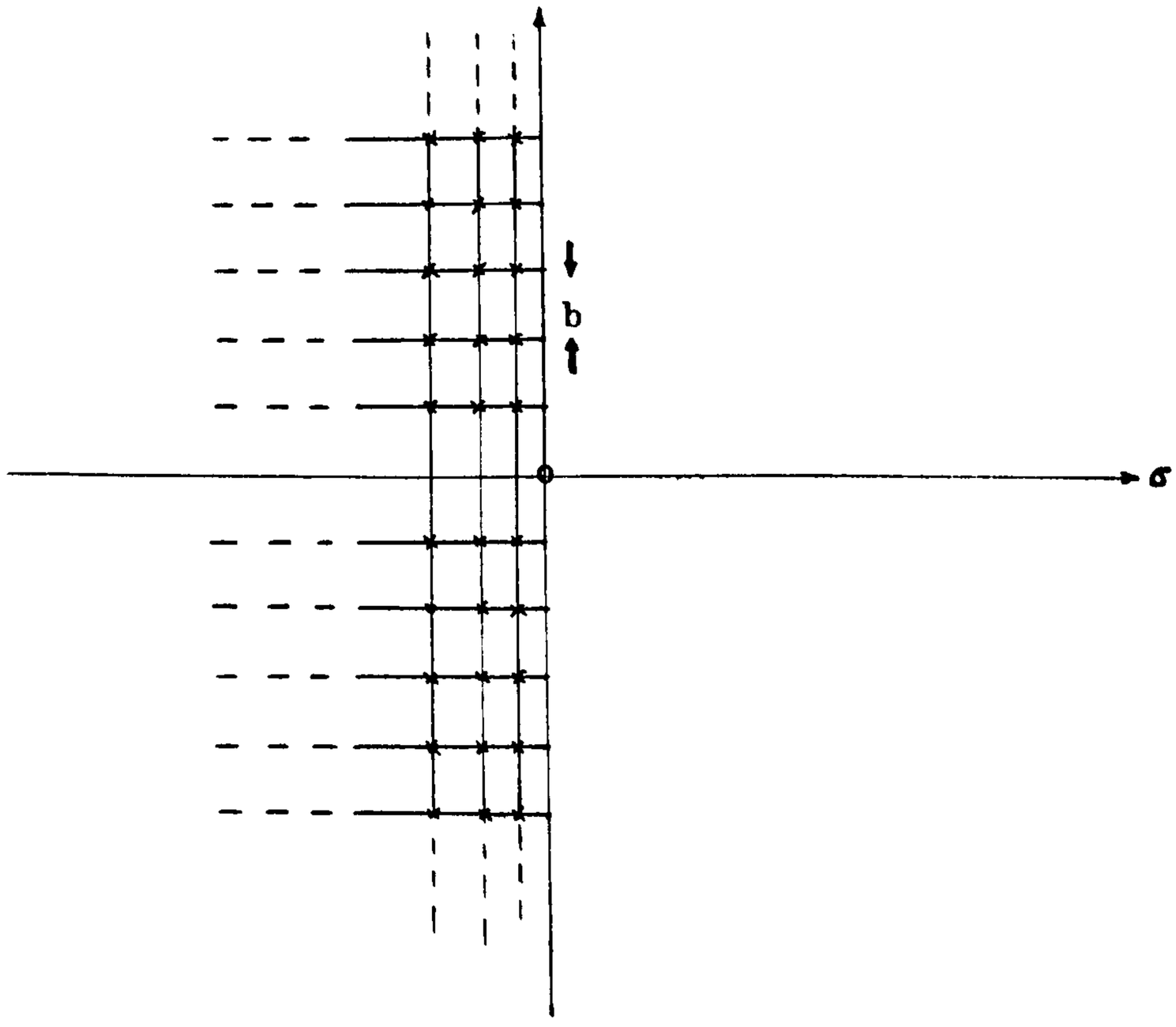


FIG: 3.2.2 S-PLANE EQUIVALENT OF THE INFINITE POLE-PATTERN IN THE Z-PLANE

Without loss of generality, equations (3.2.1) to (3.2.4) can also be written in the digital domain as follows:

$$H_a(z) = \sum_{n=-\infty}^{\infty} \frac{1-z^{-1}.e^{-aT}\cos(nbT)}{1-z^{-1}.2e^{-aT}\cos(nbT)+z^{-2}.e^{-2aT}} \qquad (3.2.5)$$

$$H_p(z) = \frac{a}{1-z^{-1}.e^{-(a+jnb)T}} \qquad (3.2.6)$$

$$H_p^*(z) = \frac{a}{1-z^{-1}.e^{-(a-jnb)T}} \qquad (3.2.7)$$

$$H_p'(z) = \frac{1-z^{-1}.e^{-aT}\cos(nbT)}{1-z^{-1}.2e^{-aT}\cos(nbT)+z^{-2}.e^{-2aT}} \qquad (3.2.8)$$

where n is an integer that varies from negative infinity to positive infinity, and the subscripts bear the usual meaning as in the analogue case.

In the derivation of the above equations, the Impulse Invariance relationship between the analogue and digital domains is assumed, although the Bilinear Z Transformation may find suitable applications in "wideband" filter designs (for a detail derivation of equations (3.2.5) to (3.2.8), the reader is now referred to appendix B). The term "wideband" is employed here to specify designs where the passbands of the resultant digital filters are an appreciable fraction of the Nyquist Interval (i.e. 70%-80% of $-\frac{1}{2}\omega_s \leqslant \omega \leqslant \frac{1}{2}\omega_s$). If the Impulse Invariance Transformation is used to derive elemental filters in the proposed technique, a guard filter is often necessary in wideband applications as explained in section 2.2.1 previously. On the other hand, if the Bilinear Z Transformation is used, due to its frequency warping characteristic, the group delay of each elemental filter has to be predistorted by a factor of $1/(1+\frac{1}{4}T^2\Omega^2)$ to provide phase linearity within the passbands of the individual elemental filters in the proposed technique in the approximation of an arbitrary frequency response. In the following discussion, the Impulse Invariance relationship is assumed. Equation(3.2.8) represents an elemental filter which is basically a high-Q resonator.

3.2.2 PHASE MODIFICATION:

We consider next the superposition of two adjacent high-Q digital resonators acting as elemental filters, and the necessary "phase modification" to approximate linear phase in the proposed design technique. From now on, we shall use the terms elemental filter and high-Q resonator interchangeably, as they refer to the same sampling "device" used in the proposed approach to sample an arbitrary frequency response.

In the proposed approach, we assume a nonminimum phase criterion and, consequently, the positions of the zeros of the elemental filters are unimportant. In fact, we can assume, without loss of generality, that the zeros of the high-Q resonators be moved to infinity such that their transfer functions are of the form:

$$H''(z) = \frac{1}{1-z^{-1}.b_1+z^{-2}.b_2} \qquad (3.2.9)$$

where the frequency response is

$$\left|H''(e^{j\omega T})\right| = \left[\frac{1}{(1-b_1\cos\omega T-b_2\cos 2\omega T)^2+(b_1\sin\omega T+b_2\sin 2\omega T)^2}\right]^{\frac{1}{2}} \qquad (3.2.10)$$

and the phase response is

$$\angle H''(e^{j\omega T}) = \tan^{-1}\left[\frac{b_1\sin\omega T+b_2\sin 2\omega T}{1-b_1\cos\omega T-b_2\cos 2\omega T}\right] \qquad (3.2.11)$$

Figure 3.2.3 depicts the frequency and phase responses of two adjacent elemental filters in the approximation technique, each being made up of a complex conjugate pole-pair as in equation (3.2.9) and resonating at frequencies ω_1 and ω_2. Provided that very high-Q poles are used (i.e. poles very close to the unit circle), the phase responses of these elemental filters can be regarded as approximately linear in the regions shown in figures 3.2.3c and 3.2.3d for the purpose of our linear phase approximation. Furthermore, it is observed that their phase relationship is approximately in phase opposition in the overlap region and almost in phase outside the overlap region. Since the responses of the two adjacent elemental filters, thus located in frequency, are such related in phase, we can, therefore, by subtracting the outputs of the two filters shown,

make the responses $H_1^n(e^{j\omega T})$ and $H_2^n(e^{j\omega T})$ effectively add during the overlap region while subtract otherwise.

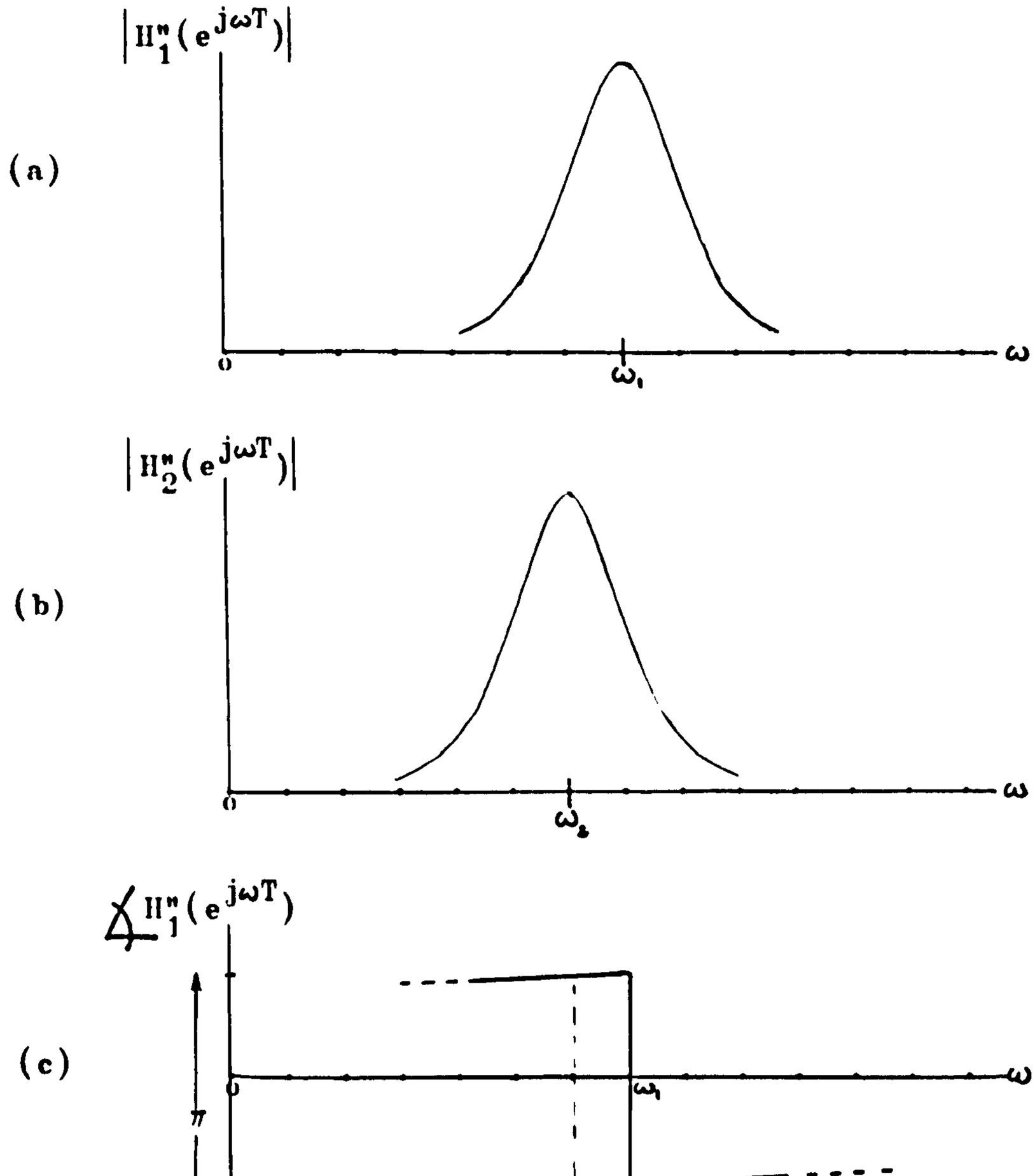


FIG: 3.2.3 FREQUENCY RESPONSES AND PHASE RELATIONSHIP OF TWO ADJACENT ELEMENTAL FILTERS

Thus, the effect of the above "phase modification" is to produce a composite frequency response (as shown in figure 3.2.4) which is fairly constant in magnitude during the overlap region of the two frequency responses. The magnitude of the passband ripple is governed by the ripple factor ϵ which will be discussed in a later section. Moreover, since these adjacent elemental filters have poles at the same distance from the unit circle, but displaced in frequency only, their phase responses are essentially the same except with a displacement in frequency. In addition, since the phase responses $\angle H_1^n(e^{j\omega T})$ and $\angle H_2^n(e^{j\omega T})$ are themselves linear, therefore, the resultant combined phase response $\angle H_{1,2}^n(e^{j\omega T})$ must also be linear with frequency in the overlap region.

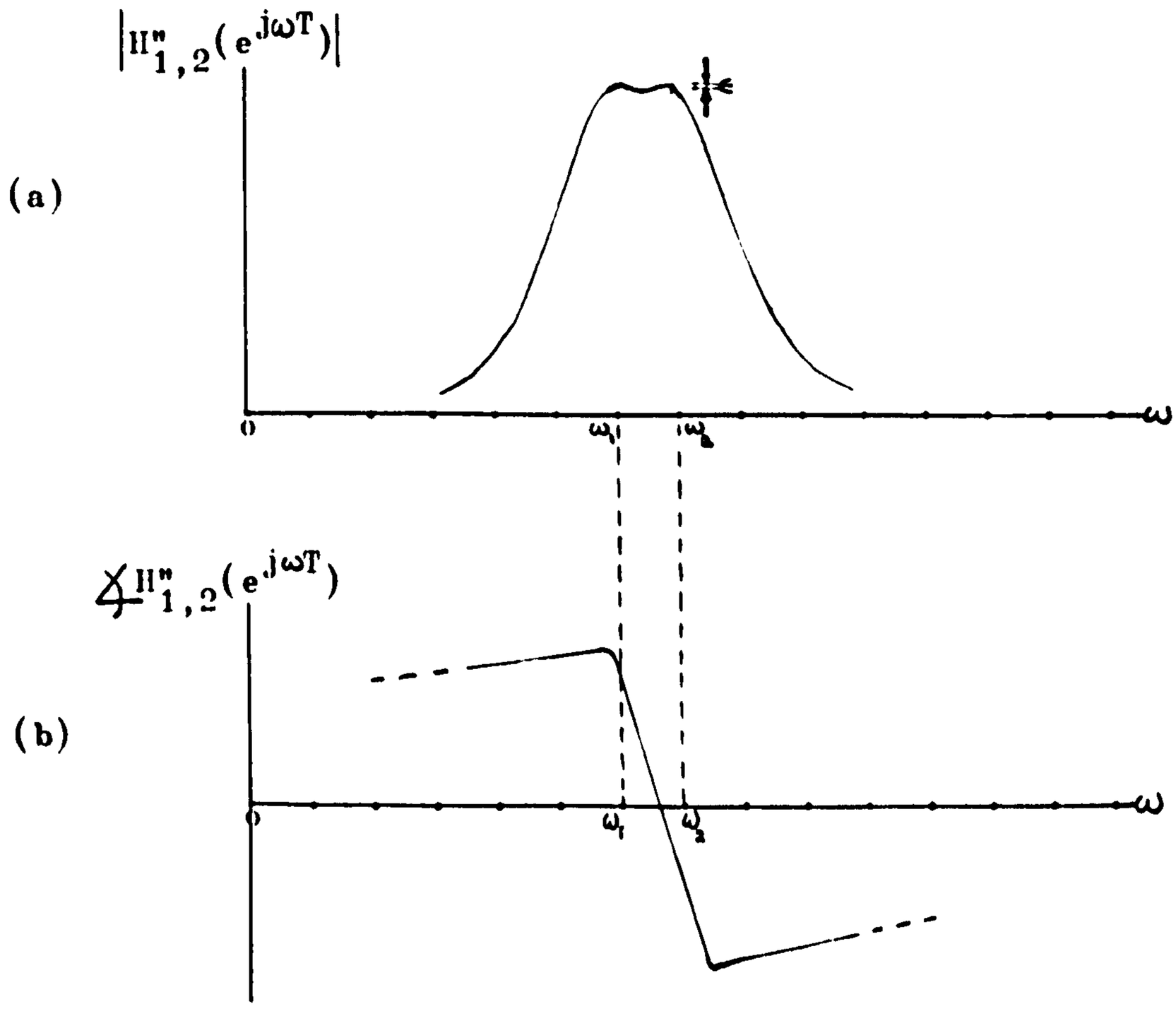


FIG: 3.2.4 THE COMPOSITE FREQUENCY AND PHASE RESPONSES OF TWO ADJACENT ELEMENTAL FILTERS COMBINED WITH "PHASE MODIFICATION".

Extending the above argument to the case of "n" elemental filters, we can achieve the first step of our approximation technique by having produced an effective frequency response that resembles the adding of true interpolation sinc functions, $\frac{\sin x}{x}$, as in the frequency sampling design. Therefore, an arbitrary filter spectrum with approximately linear phase over the entire spectrum of interest, can be specified by sampling the desired passband at selected equispaced sampling instants, using the above elemental filters.

Figure 3.2.5 depicts the whole scene when phase modification is applied to the poles on the concentric circles within the unit circle in the infinite structure in the z-plane shown in figure 3.2.1.

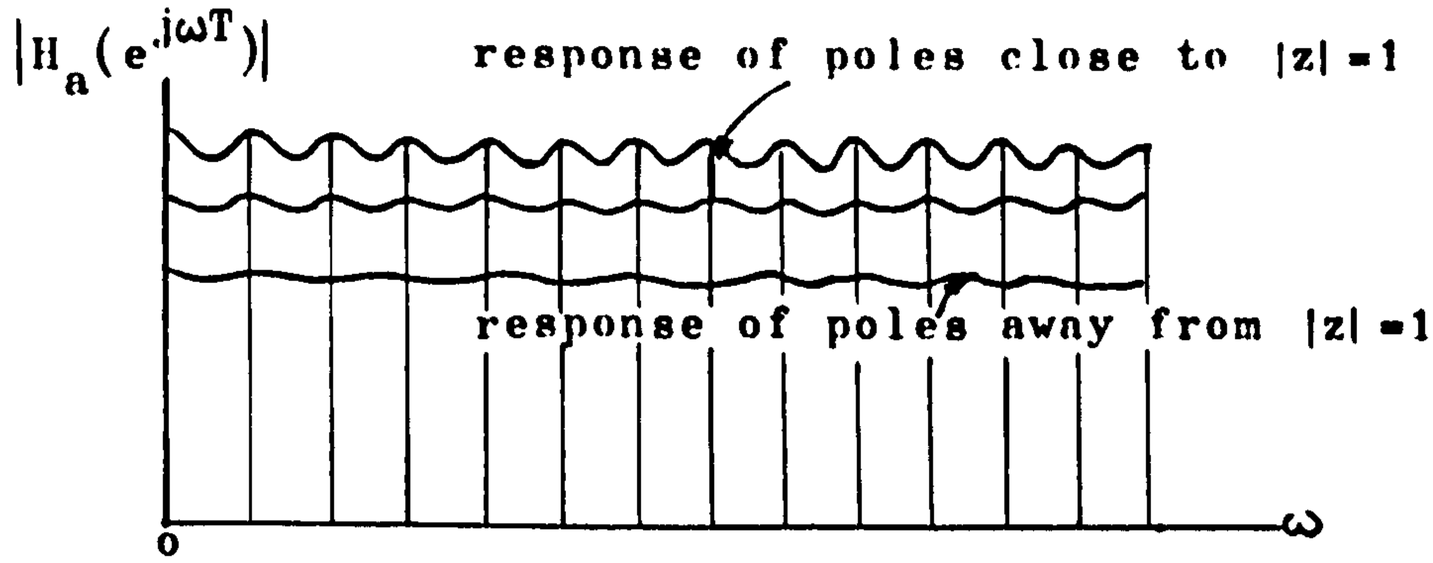


FIG: 3.2.5 FREQUENCY RESPONSES OF POLES ON CONCENTRIC CIRCLES WITHIN THE UNIT CIRCLE FOR VARIOUS VALUES OF R IN THE Z-PLANE INFINITE STRUCTURE SHOWN IN FIGURE 3.2.1.

Hence we observe that the passband ripple and the amplitude of the frequency response at any frequency are increasing with the value of R being close to unity, or equivalently with decreasing value of a, the distance of poles from the $j\Omega$-axis in the s-plane grid. However, the magnitude of the passband ripple is not simply governed by the value of R or a alone, and we shall see in the next section that it also depends on the angular separation i.e. the angle bT, between poles in the z-plane. This implies that the magnitude of the passband ripple also depends on b, the separation between poles in the s-plane grid, in addition to the value of a.

3.2.3 RIPPLE ANALYSIS:

After phase modification, equation(3.2.1) becomes

$$H(s) = \sum_{n=-\infty}^{\infty} \frac{(-1)^n.a}{s+(a-jnb)} \qquad (3.2.12)$$

Similarly, the z-plane transfer function will become

$$H(z) = \sum_{n=-\infty}^{\infty} \frac{(-1)^n(1 - z^{-1}.e^{-aT}\cos(nbT))}{1 - 2.e^{-aT}\cos(nbT)z^{-1} + e^{-2aT}.z^{-2}} \qquad (3.2.13)$$

Although it is always possible to analyse the ripple factor ϵ in the z-domain, we have chosen here, without loss of generality, to analyse ϵ in the s-domain. The frequency response $H(j\Omega)$ of an infinite number of poles with separation b apart, and all lying on an axis parallel to and distant a from the $j\Omega$-axis, is shown in figure 3.2.6 below:

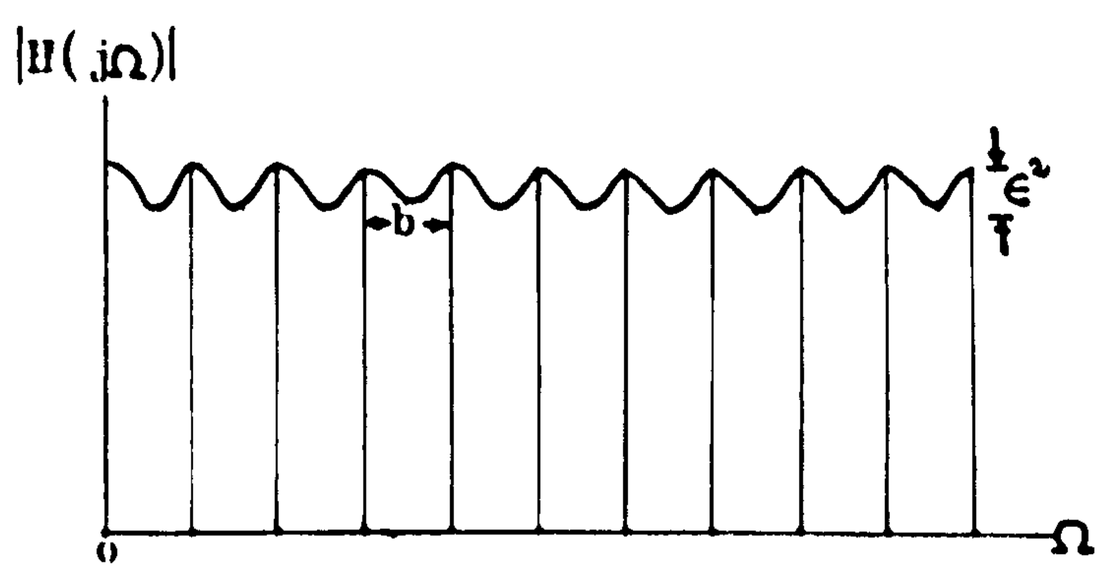


FIG: 3.2.6 PASSBAND RIPPLE

Now from equation(3.2.2), we have the single-pole response contributes towards that of H(s) in equation(3.2.12) as

$$H_p(j\Omega) = \frac{a}{j(\Omega-nb)+a} \qquad (3.2.14)$$

Its real part is

$$\mathrm{Re}: H_p(j\Omega) = \frac{a^2}{(\Omega-nb)^2+a^2} \qquad (3.2.15)$$

Equation(3.2.15) shows that the real part of the response of a single pole along the jΩ-axis roughly resembles a half-cosine wave in the neighbourhood of Ω=nb, as shown in fig.3.2.7. Such a single-pole response is used with phase modification in our linear phase approximation technique.

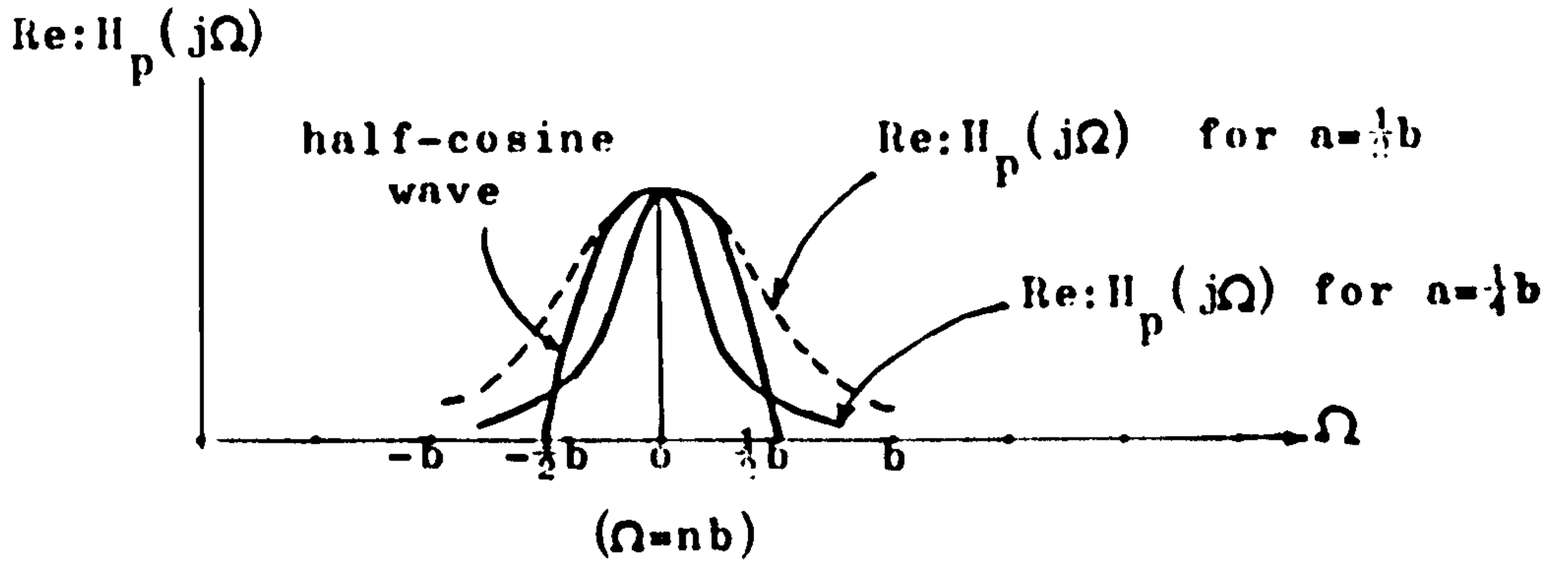


FIG: 3.2.7 REAL PART APPROXIMATION OF H(s)

The frequency response of the infinite number of poles on the parallel axis with phase modification can be written by substituting jΩ for s in equation(3.2.12) as follows:

$$H_p(j\Omega) = \sum_{n=-\infty}^{\infty} \frac{(-1)^n . a}{j(\Omega - nb) + a} \qquad (3.2.16)$$

which represents a ~~good~~ approximation to a function whose real part is $\cos(\frac{\pi\Omega}{b})$ on the jΩ-axis, which is periodic in Ω with a period equal to 2b.

Furthermore, equation(3.2.16) can be summed in closed form (37),(38),(39), in terms of hyperbolic functions as

$$H_p(j\Omega) = \frac{\pi a/b}{\sinh(\pi a/b + j\pi\Omega/b)} \qquad (3.2.17)$$

which can be developed into a series form in Ω by the binomial expansion as

$$H_p(j\Omega) = \frac{2\pi a}{b}\left(e^{-\pi a/b}.e^{-j\pi\Omega/b} + e^{-3\pi a/b}.e^{-3j\pi\Omega/b} + e^{-5\pi a/b}.e^{-5j\pi\Omega/b} + \ldots\ldots\right)$$

so that, $$H_p(j\Omega) = \frac{2\pi a}{b}(\epsilon.e^{-j\pi\Omega/b} + \epsilon.e^{-3j\pi\Omega/b} + \epsilon.e^{-5j\pi\Omega/b} + \ldots.) \quad (3.2.18)$$

where the symbol e has been designated as the ripple factor of the above infinite passband, and

$$\epsilon = e^{-\pi a/b} \quad (3.2.19)$$

The magnitude of the infinite structure H(s) in equation (3.2.12) is then given by an approximate expression below:

$$|H(j\Omega)| \simeq \frac{2\pi a\epsilon}{b}(1 + \epsilon^2.\cos(\frac{2\pi\Omega}{b}) + \ldots\ldots) \quad (3.2.20)$$

Thus, H(s) approximates a constant frequency response with a periodic passband ripple of relative magnitude given by

$$\epsilon^2 = e^{-2\pi a/b} \quad (3.2.21)$$

Now, ϵ^2 is designated as the ripple magnitude, and from equation (3.2.21), it is seen that the above approximation is equiripple with a constant magnitude for a given pair of a and b. However, if this infinite passband is truncated into finite passbands and stopbands, then the above approximation may not be equiripple at the bandedges due to the truncation of the infinite passband structure. This truncation results in a general design model which will be described in the next section. If we now put

$$\lambda = \frac{2a}{b} \quad (3.2.22)$$

and call it the ripple ratio, we observe that by making λ sufficiently large, ϵ^2 may be made as small as we please. For instance, for λ=1, the amplitude error in the approximation is $\pm$5%. Nevertheless, the foregoing analysis has provided important equations which link up the essential design parameters, a and b, which represent the distance of the poles from the $j\Omega$-axis and their separation respectively.

3.2.4 TRUNCATION OF THE INFINITE PASSBAND INTO FINITE PASSBANDS- A GENERAL DESIGN MODEL:

Having found the transfer function for an infinite passband approximating a constant amplitude response in frequency, we now proceed to truncate this infinite structure into finite passbands and stopbands. This is the next part of our approximation procedure where we drop all poles in the stopband, leaving only poles in the passband of the desired frequency response.

However, dropping all poles in the stopband in a straightforward manner will result in a response with a slow transition characteristic and little out-of-band rejection or attenuation, which may not be desirable in most circumstances. In the proposed approach, optimization is used as a remedy (the optimization procedure used will be described in section 3.2.6). For this purpose, a transition band is introduced between the passband and the stopband, wherein lie one or more frequency samples which are being made into transition poles. These transition poles, each of which corresponds to an elemental filter in our realization, are spaced at equal distance to sample the entire transition band. The resultant transition samples are made into unconstrained varinbles in the optimization procedure in order to optimize our design.

As mentioned previously, these optimally weighted transition samples, though equispaced, are seperated by a distance less than that of the passband samples. Consequently, this greatly improves the flexibility of placing the transition samples, but suffer, on the other hand, a phase nonlinearity at the bandedge which will otherwise be linear in the approximation. In addition, owing to the shorter seperation between transition samples, the transition band has become narrower than what would have been if the same separation between passband samples had been used. As a result, the slope of the transition skirt becomes steeper. This effect, when coupled with the use of optimization will enhance the overall frequency response of the resultant design, giving results much more satisfactory than a simple, straightforward truncation of the infinite structure.

We shall now describe a general design model for the proposed approach whereupon a suitable optimization procedure will be applied.

Without loss of generality, the model shall be laid down for a bandpass structure, whence other important designs such as lowpass filters etc., can be similarly derived. We also define in our model, tolerance sets and error sets for the sake of satisfying the specifications of desired frequency responses.

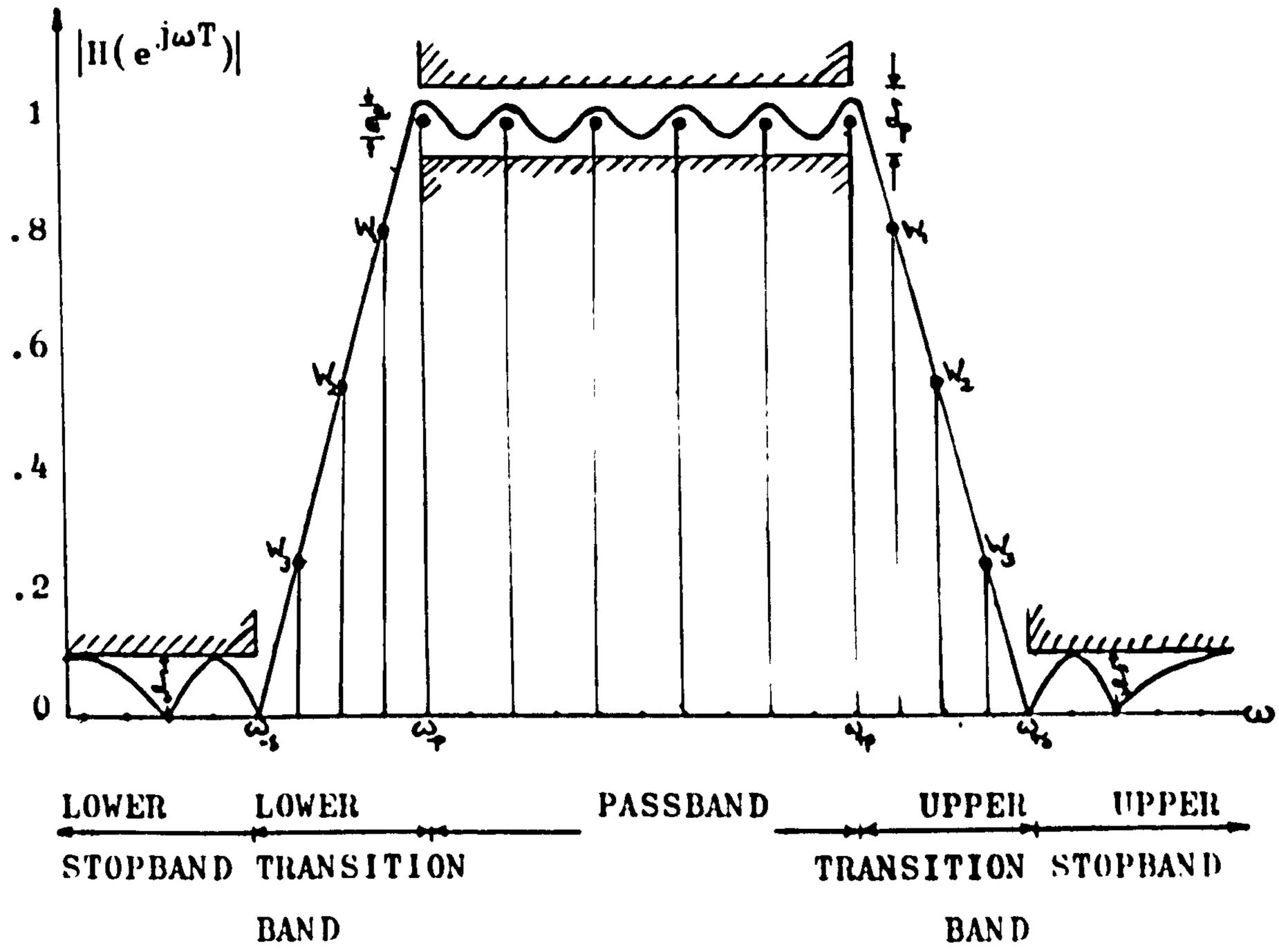


FIG: 3.2.8 A GENERAL DESIGN MODEL

Figure 3.2.8 shows a tolerance scheme of a bandpass filter described by the desired frequency response,

$$\left|H_p(e^{j\omega T})\right| = \begin{cases} 1 \text{ in the passband.} \\ 0 \text{ in the stopband.} \end{cases} \qquad (3.2.23)$$

with tolerance set,

$$\delta(\omega) = \begin{cases} \delta_p \text{ in the passband.} \\ \delta_s \text{ in the stopband.} \end{cases} \qquad (3.2.24)$$

A transfer function $H(z)$ satifies the tolerance set if,

$$\left|\,|H_D(e^{j\omega T})| - |H(e^{j\omega T})|\,\right| \leq \delta(\omega) \qquad \forall\omega \qquad (3.2.25)$$

where the left hand side of the inequality is the error set. Define

$$E(\omega) = \frac{1}{\delta(\omega)}\left|\,|H_D(e^{j\omega T})| - |H(e^{j\omega T})|\,\right| \leq 1 \qquad \forall\omega \qquad (3.2.26)$$

where $E(\omega)$ is seen to be a figure of merit, and for every satisfactory design, equation(3.2.26) holds. In general, we have

$$E(\omega) = \begin{cases} \frac{1}{\delta_p}\left|\,|H_D(e^{j\omega T})| - |H(e^{j\omega T})|\,\right| & \text{for } \omega_p<\omega<\omega_p \\ \frac{1}{\delta_s}\left|\,|H_D(e^{j\omega T})| - |H(e^{j\omega T})|\,\right| & \text{for } 0<\omega<\omega_s \wedge \omega_s<\omega<\frac{\pi}{T} \end{cases} \qquad (3.2.27)$$

Since, once $\{\omega_s, \omega_s, \omega_p, \omega_p\}$ are chosen, then there are no constraints in the transition bands between the passband and the stopbands, and $E(\omega)$ is not defined in these regions. However, $E(\omega)$ can also be defined in the transition bands if we further impose the constraint that the slope of the transition skirt should be as steep as possible in the optimized response. In this case, the achievable steepness of the transition skirt varies inversely with the out-of-band attenuation attainable. In general, the wider the transition bands (i.e. less steep slope), the larger the out-of-band attenuation and vice versa. By minimizing the error set in the approximation technique, we obtain a set of coefficients of a filter function with equiripple passband behaviour which satisfies $\delta(\omega)$. Also, a certain margin which can be used for the improvement of $H(e^{j\omega T})$ in some respects exists, when the value of $E(\omega) < 1$.

Furthermore, in the above model, we can define the following:

N = number of passband poles,

M_u = number of poles in the upper transition band,

M_l = number of poles in the lower transition band,

K = total number of poles to be realized,

$= N + M_u + M_l$

W_n = weighted transition samples in the optimization process.

3.2.5 THE EFFECT OF TRANSITION POLE-PAIRS:

Before discribing the optimization process used in the proposed approach, we now further discuss the effect of introducing transition samples in the transition band. In the case of the bandpass model, these transition samples are in fact transition pole-pairs.

Prior to optimizing the various optimization weights W_n, they are first empirically decided as shown in figure 3.2.8. For example, $\{W_1, W_2, W_3\}$ in the above bandpass model would empirically be chosen as $\{.8, .54, .22\}$ respectively. However, this set of values of W_n is only one of the many possible sets of values of W_n, and therefore, has to be optimized to find the most optimal or feasible solution for a given design.

Furthermore, the seperation of these transition samples is proportional to the ripple ratio λ. In the proposed approach, the separation of the equispaced transition samples is taken as sub-multiples of that between the passband samples. The choice of making the above seperation as sub-multiples is primarily based on the modularity and enhancement of the resultant design. In particular, this choice has made possible a pole-sharing property in the transition regions which will be discussed in detail in section 3.4.

Figure 3.2.9 shows the responses of two designs by truncating the infinite structure of figure 3.2.6 into a finite bandpass spectrum. Figure 3.2.9a shows the response of a 4-pole bandpass filter design without any transition samples, while figure 3.2.9b depicts the design of a 4-pole bandpass filter with the introduction of a pair of transition poles. In figure 3.2.9a we have the passband poles separated by b in frequency, whereas, in figure 3.2.9b, the separation of the passband poles is 2b and that of the transition poles is simply b. Furthermore, both designs have N=4, however, in figure 3.2.9a, $M_u = M_l = 0$ while, $M_u = M_l = 1$ in figure 3.2.9b.

From figure 3.2.9, it is obvious that the introduction of the transition pole-pair has the effect that the stopband attenuation is significantly improved. That this is feasible is due to the fact that the transition bands are now under the influence of the weighted transition samples as opposed to the uncontrolled case in fig. 3.2.9a.

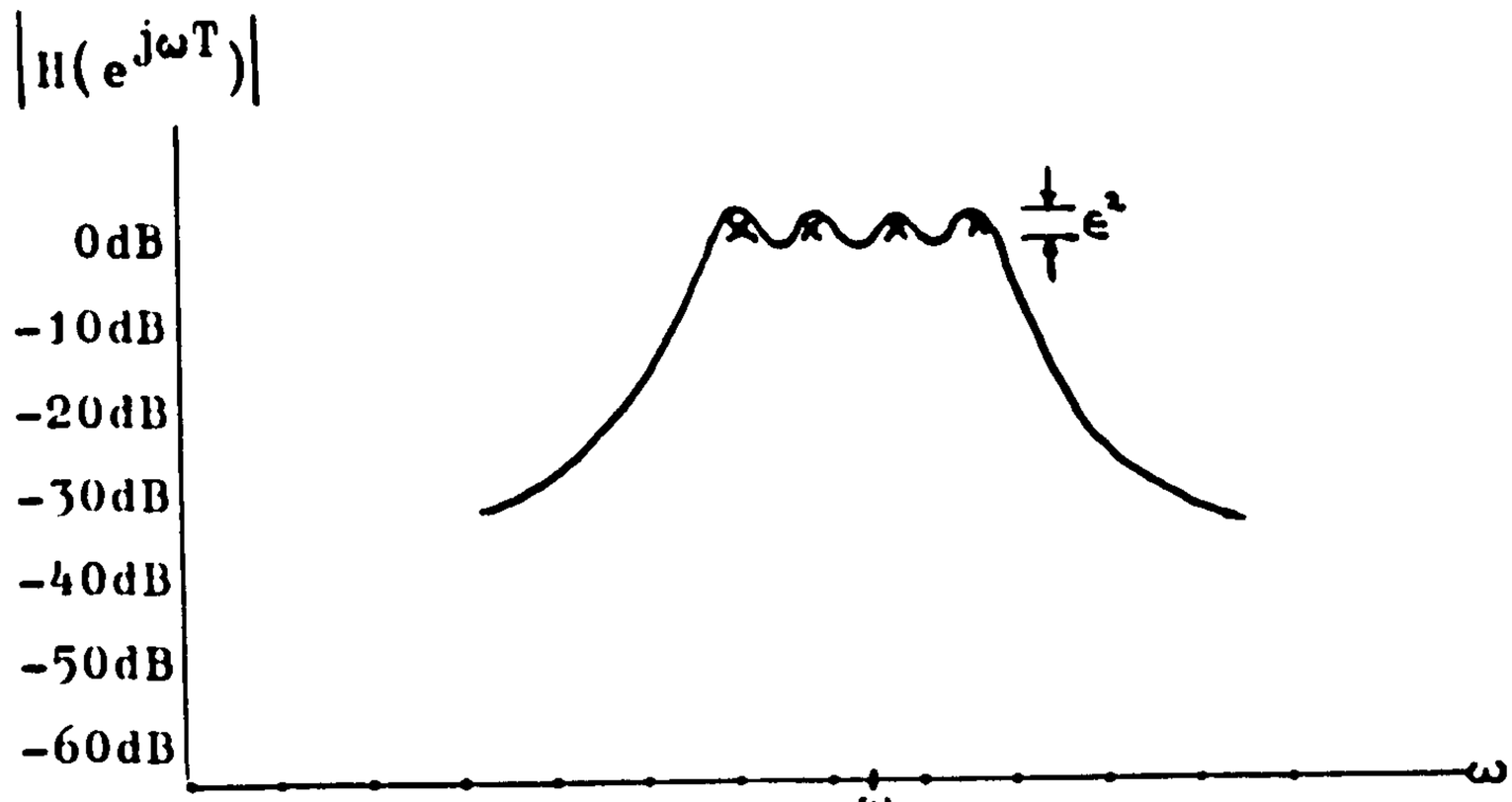


FIG: 3.2.9a 4-POLE BANDPASS RESPONSE WITHOUT TRANSITION POLES

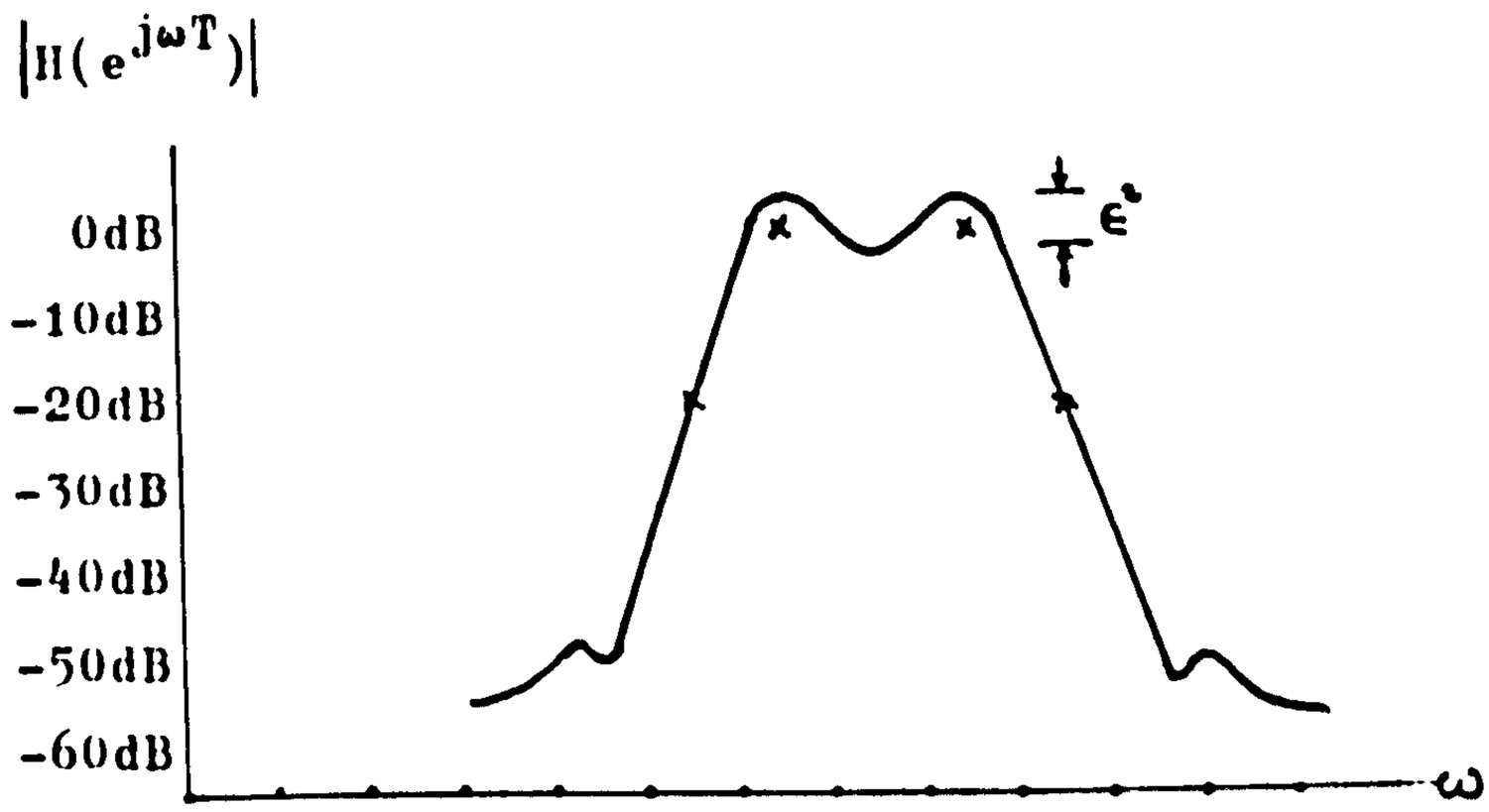


FIG: 3.2.9b 4-POLE BANDPASS RESPONSE WITH A PAIR OF TRANSITION POLES

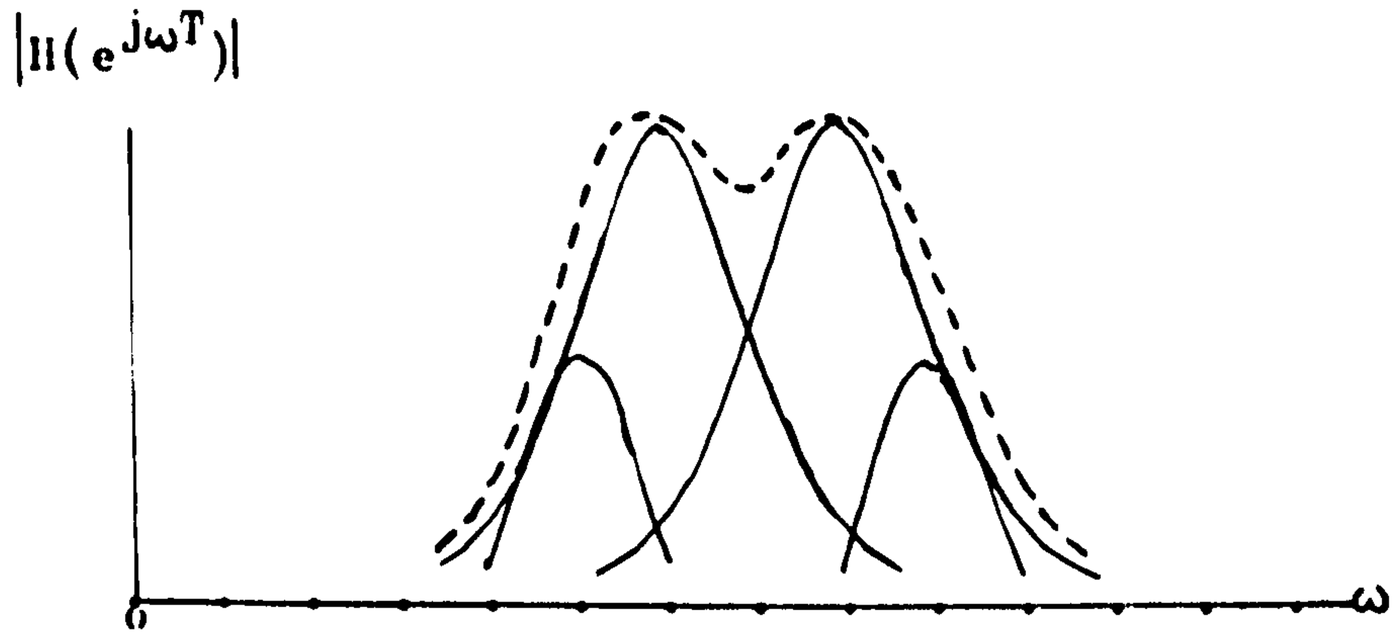
In essence, the transition samples are seen to behave as interpolation samples in the transition bands to enable a smoother transition. By weighting these samples optimally, the characteristic of the frequency response in the transition bands and the stopbands varies accordingly. Again, the optimal result thus obtained, is in almost all circumstances, much better than that by merely using empirical weights for the transition samples.

Extending the above argument further, we can broaden the transition bands by introducing more transition pole-pairs to obtain even greater out-of-band attenuation. For instance, if the number of passband poles, N, is fixed, then the transition bands will be

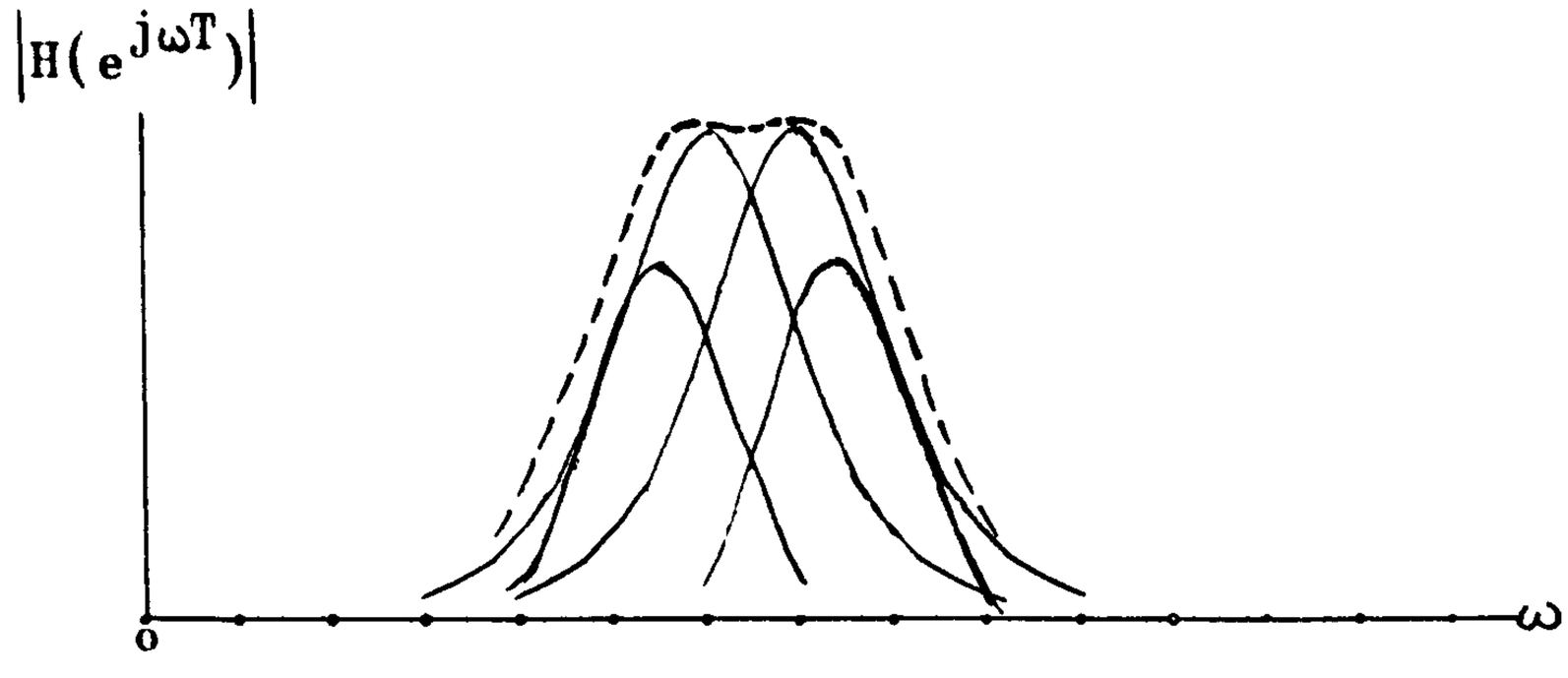
twice as wide, with the introduction of an extra pair of transition poles, properly weighted. On the other hand, increasing the number of passband poles i.e. N, for a given passband, while keeping the number of transition samples fixed, gives better stopband attenuation. This also allows a narrower transition band due to a decreased seperation between both the passband and transition band poles. Furthermore, one can always use higher-order poles i.e. higher-order elemental filters, in the transition band to optimize the design so that on a whole, the transition band will be steeper and narrower than when simple poles are used. For a description of higher-order elemental filters, the reader is referred to appendix B.

In addition to the above, it is noticed that the passband ripple in figure 3.2.9b is larger than that of figure 3.2.9a (the magnitude of the ripples have been exaggerated in both cases for clarity). This is due to the fact that the separation of the passband poles in figure 3.2.9a is half that of figure 3.2.9b, thus the ripple ratio λ is halved consequently. We further notice that N=2 in figure 3.2.9b while N=4 in figure 3.2.9a, which has thus made all the difference.

Figure 3.2.10 depicts the variation of the passband ripple magnitude ϵ^2, due to a variation in the value of b for a fixed value of a in λ, and a given value of N=2, in a 4-pole bandpass design.



(a) PASSBAND RIPPLE MAGNITUDE DUE TO A WIDE SEPARATION "b" OF THE PASSBAND POLES



(b) PASSBAND RIPPLE MAGNITUDE DUE TO AN OPTIMUM SEPERATION "b" OF THE PASSBAND POLES

FIG: 3.2.10 VARIATION OF ϵ^2 DUE TO A VARIATION IN "b"

In short, one generally chooses the appropriate or optimum parametric values of a and b, to meet the specification of a desired frequency response.

3.2.6 OPTIMIZATION:

Having dealt with the effect of the transition poles, we are now in a position to describe the final step of our approximation technique: optimization.

The general transfer function to be optimized, representing the bandpass model shown in figure 3.2.8, can be written as

$$H(z) = \sum_{n=1}^{M_1} \frac{W_n(-1)^{M_1-n}(1 - z^{-1}.e^{-aT}\cos((\omega_p - nb)T))}{1 - z^{-1}.2e^{-aT}\cos((\omega_p - nb)T) + z^{-2}.e^{-2aT}} +$$

$$\sum_{k=o}^{N-1} \frac{(-1)^{M_1+k}(1 - z^{-1}.e^{-aT}\cos((\omega_p + 2kb)T))}{1 - z^{-1}.2e^{-aT}\cos((\omega_p + 2kb)T) + z^{-2}.e^{-2aT}} +$$

$$\sum_{n=1}^{M_u} \frac{W_n(-1)^{M_1+N+n+1}(1 - z^{-1}.e^{-aT}\cos((\omega_{vp} + nb)T))}{1 - z^{-1}.2e^{-aT}\cos((\omega_{vp} + nb)T) + z^{-2}.e^{-2aT}} \qquad (3.2.28)$$

where the separation between passband poles is 2b and that between transition poles is b.

The optimization procedure used in the proposed approach is a modified version of the "Gradient Method" (29), (40)-(43), in which the principle of "Steepest Descent" is employed. The method of gradients or steepest descent is intuitively attractive and has had a long history of application. Cauchy referred to it in an 1847 publication in connection with the solution of simultaneous equations. The thought behind the method is quite simple: if one starts at any point and goes continuously downhill, a minimum must eventually be reached.

In essence, the optimization process of the proposed linear phase approximation technique is the finishing touch which refines individual designs by optimizing the set of parameters $\{a, b, W_n\}$ to suit the tolerance set $\delta(\omega)$. In the process, the error set is minimized, while the transition slope $\frac{d}{d\omega}|H_T(e^{j\omega T})|$ is maximized, for minimum passband ripple and steepest transition skirt coupled with a sufficient amount of out-of-band attenuation to satisfy a given specification.

This process is in fact a (n+2)-dimensional search problem as there are (n+2) parameters in all. The process is best carried out with an interactive on-line computer programme, with graph-plotting facilities, so that the designer can vary the paramters with visual aids from an instantaneous plot of the frequency response. In the latter part of this section, we shall describe an N-dimensional search programme suitable for such purposes.

Experience has shown that by first calculating the magnitude of the passband ripple ϵ^2 by equation(3.2.21) to satisfy the tolerance set δ_p in equation(3.2.24), we can reduce the number of parameters to n i.e. $\{W_n\}$ and the whole problem simply becomes an optimization of the set of transition samples $\{W_n\}$ to satisfy δ_s and to maximize $\frac{d}{d\omega}|H_T(e^{j\omega T})|$. The out-of-band attenuation achievable in a particular design depends on the parameter a and the number of transition samples used. The ripple ratio λ can specify the value of b for a given a, to suit δ_p in equation(3.2.24) which in turn dictates the number of passband poles N required. Whenever a set of $\{a, b\}$ fails to meet specification of a desired frequency response, even after optimizing the weights $\{W_n\}$, one then simply change to a "tighter" set of $\{a, b\}$. However, the

optimization of $\{W_n\}$ is seen to be more important in most cases than that of the set $\{a, b\}$ which can be regarded as a separate problem in this instance. In deciding the number of transition samples required, one simply starts off by inserting one pair of transition poles in the transition bands on both sides of the passband, and increasing the number of transition pole-pairs to achieve the required attenuation, while changing the values of the set $\{a, b\}$ if necessary. Experience has shown that the process of optimizing $\{W_n\}$ may introduce small variations in the precalculated value of the passband ripple ϵ^2, however, these variations can often be easily adjusted.

The optimization procedure can be broadly divided into three seperate steps as follows:

STEP ONE: We start off with a one dimensional search for a given design by varying W_1 between the valuss of 0 and 1. An empirical value of W_1 is first found between 0 and 1 as an initial value for the search programme. This will apply in general to the whole set of $\{W_n\}$ whence we start optimizing from a set of empirical values which may in some cases vary quite drastically from the final optimum values. Figure 3.2.11 shows the variation of $\frac{d}{d\omega}|H_T(e^{j\omega T})|$ through the optimum value of W_1 in the one dimensional search.

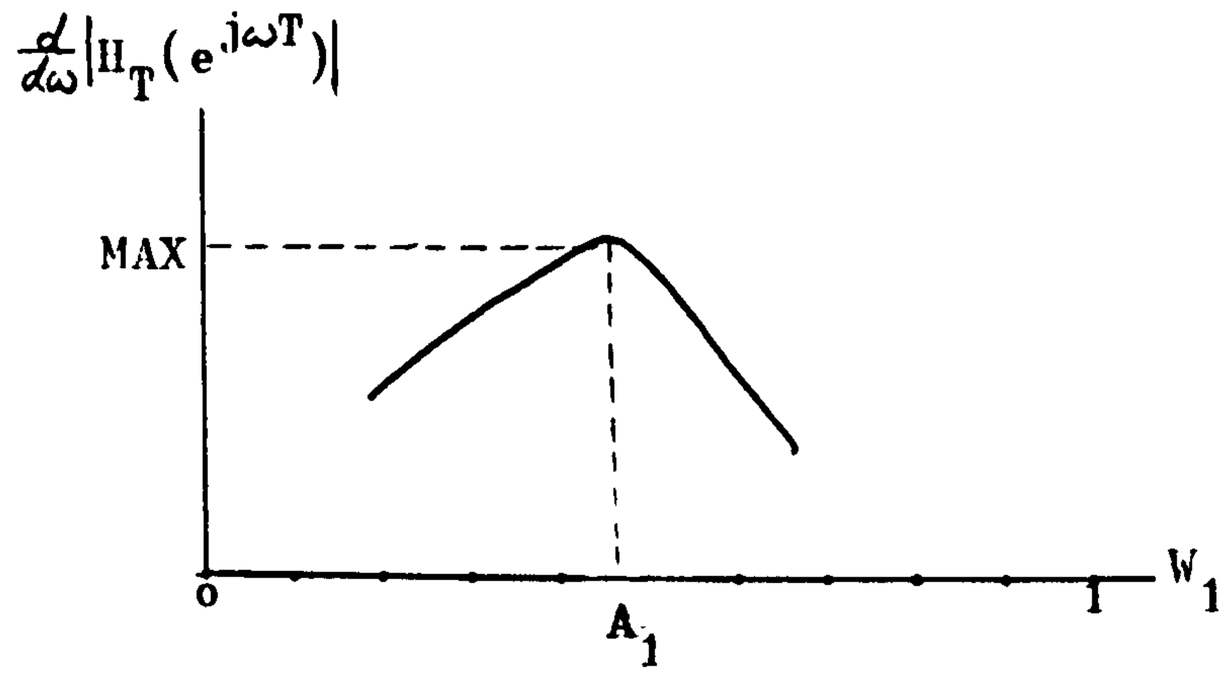


FIG: 3.2.11 ONE DIMENSIONAL SEARCH ON THE W_1-LINE

We stop at finding the optimum value A_1 for the transition sample W_1. If the response at this stage does not satisfy

the desired specifications, for instance, a lack of out-of-band attenuation, we can then try a "tighter" set of a, b and repeat the whole one dimensional search. Failing this, we then go on to two transition samples and follow a two dimensional search in step two below.

STEP TWO: We begin the two dimensional search by first plotting W_1=1 and W_2=A_1 in a graph of W_1 vs W_2. To find another point in this W_1-W_2 plane, we decrease W_1 from its preset value of unity to a slightly smaller fraction, and repeat the one dimensional search by varying W_2 as in step one until we reach the point A_2 in figure 3.2.12 on the A-line as shown. This A-line is then our first "line of steepest slope" along which we perform our two dimensional search.

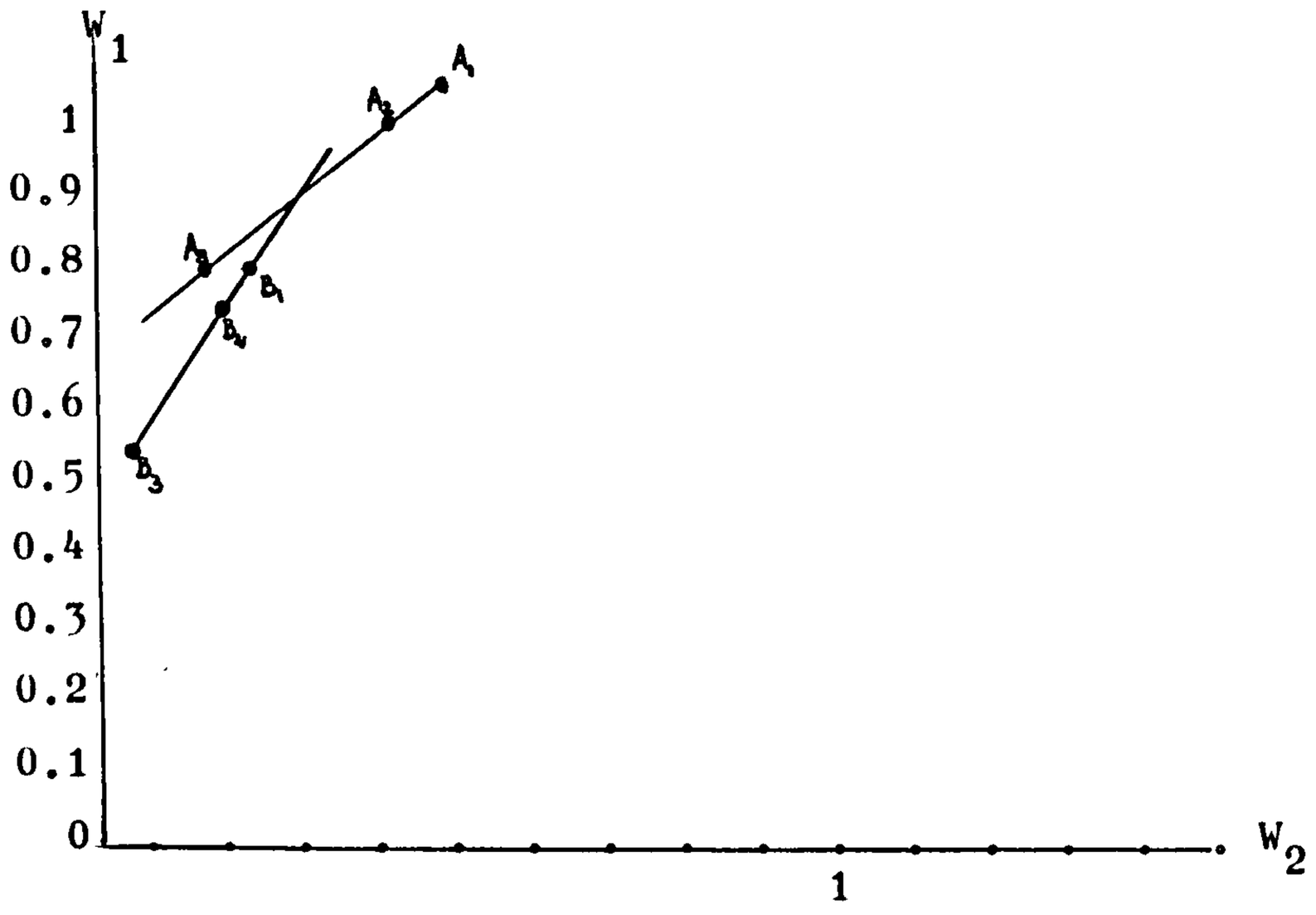


FIG: 3.2.12 TWO DIMENSIONAL SEARCH FOR THE OPTIMUM VALUES OF W_1 AND W_2 IN THE W_1-W_2 PLANE

Having defined our A-line in the W_1-W_2 plane, we now perform a simple search using the values of points along the line until we eventually arrive at the point A_3 where $\frac{d}{d\omega}|H_T(e^{j\omega T})|$ is once again optimum. Now a new line of steepest slope can be found by keeping W_1 fixed at the

value A_3 and linearly varying W_2 till we find B_1 on the B-line; then perturbing W_1 from this value and varying W_2 as before to give the point B_2. As before, perform a simple search along this B-line until we come to B_3 where $\frac{d}{d\omega}|H_T(e^{j\omega T})|$ is once again optimum. Now, if the difference between the optimum values of $H(z)$ at A_3 and B_3 is negligible, or falls within a certain design threshold, then the two dimensional search has come to an end with B_3 as solution, otherwise, iterating the same process above as before, until the ultimate optimum solution is reached.

STEP THREE: Finally, if further attenuation is required, one has to resort to three transition samples, necessitating a three dimensional search in the W_1-W_2-W_3-space. The three dimensional search follows closely the reasoning of the one dimensional and two dimensional searches. By analogy to the one and two dimensional searches, we first set W_1 to unity and with the previous solution B_3 for W_2 and W_3, we plot a point in a three dimensional W_1-W_2-W_3-space as shown in figure 3.2.13. To find another point in this space, we perturb the sample W_1 from its preset value of unity, and repeat steps one and two above. Having fixed two points in space, we can draw a three dimensional line along which we search for our optimum solution for approximating $H(z)$. From the points A_1, A_2, A_3 on the A-line, we again produce as before a second B-line and repeat all the above steps until we reach our new optimum solution on this line as B_3. This three dimensional search ends when solutions on consecutive lines have a negligible difference, otherwise, the search continues.

The above optimization procedure can obviously be extended into an N-dimensinnal search procedure for optimizing N variables. However, the procedure will be most efficient for filter designs with up to four transition samples in their transition bands, and in practice, one rarely requires more than three transition samples to meet a desired specification.

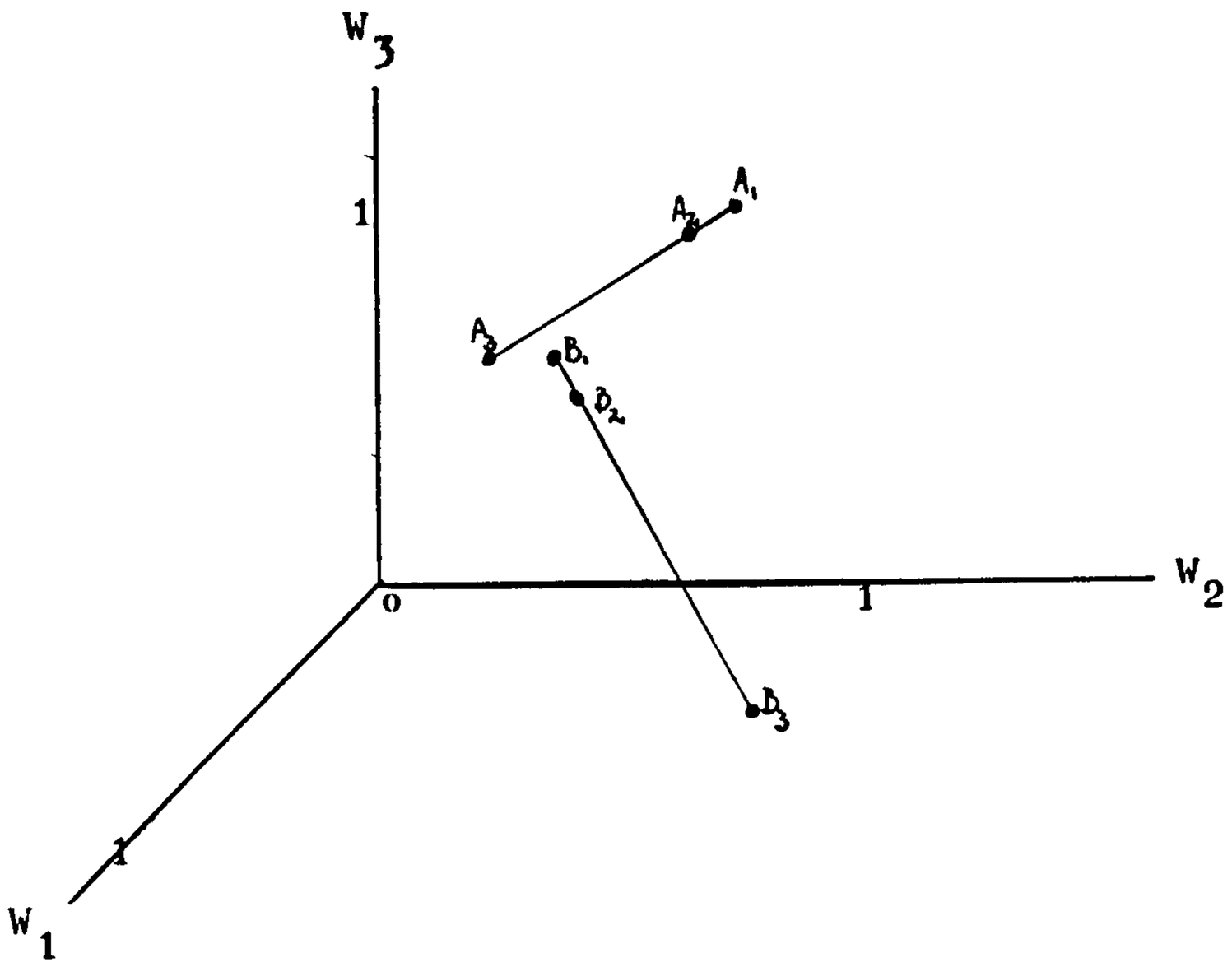


FIG: 3.2.13 THREE DIMENSIONAL SEARCH FOR THE OPTIMUM VALUES OF W_1, W_2 AND W_3 IN THE W_1-W_2-W_3-SPACE

Moreover, this search procedure converges rapidly for a single pair of transition poles for the bandpass model, corresponding to a simple one dimensional search. Often we find that if more computing time is available, a few dB's of extra stopband attenuation is achievable. Thus there is always the inevitable tradeoff between computing time and performance.

Chapter eight shows examples of actual designs of some 4-pole bandpass filters where passband ripples are apparent. In designs where the transition skirt is more gradual, the stopbands do not exhibit out-of-band ripples. However, when the designs are further optimized for the steepest transition skirt possible, then out-of-band ripples become apparent.

So far, the general design model has been relevant to bandpass filters, however, as previously mentioned, when other designs are required, suitable modifications should then be made accordingly. For instance, a lowpass design will involve dropping all poles in in the infinite structure beyond the upper transition band. The

same tolerance set and error set can also be defined similarly, while the above optimization procedure still applies except that we are dealing with individual transition samples and not pole-pairs as in the bandpass case. Equation(3.2.28) is correspondingly modified to become

$$H_L(z) = \sum_{k=1}^{N} \frac{(-1)^k(1-z^{-1}.e^{-aT}\cos(2kbT))}{1-z^{-1}.2e^{-aT}\cos(2kbT)+z^{-2}.e^{-2aT}} + \sum_{n=1}^{M_u} \frac{W_n(-1)^{N+n}(1-z^{-1}.e^{-aT}\cos((\omega_p+nb)T))}{1-z^{-1}.2e^{-aT}\cos((\omega_p+nb)T)+z^{-2}.e^{-2aT}} \qquad (3.2.29)$$

3.3 ZERO-SHARING PROPERTY:

In equation(3.2.28), we observe that the coefficient of the zero in the numerator of each elemental filter is exactly half the value of the coefficient of the z^{-1} term in the denominator, which produces the poles of the filter. Thus, in the filtering process, only two multiplications are required to generate an output term in each second-order section, as opposed to four multiplications required in a general second-order section, where both the numerator and the denominator are quadratic. However, this is not all the savings to the proposed design technique. In this section, we will further point out alternative forms of equation (3.2.28) for the general design model leading to a zero-sharing property of the proposed approach.

We recall from general system theory that the positions of zeros in a transfer function have only effects in the stopband region, while the passband characteristics are mainly governed by the positions of the poles instead. Hence, we can move the positions of the zeros of the elemental filters arbitrarily, provided that they do not fall within the passband. This suggests an alternative form of equation(3.2.28) as follows:

$$H(z) = \sum_{n=1}^{M_1} \frac{W_n(-1)^{M_1-n}(1-z^{-1}.e^{-aT})}{1-z^{-1}.2e^{-aT}\cos((\omega_p-nb)T)+z^{-2}.e^{-2aT}} +$$

$$\sum_{k=o}^{N-1} \frac{(-1)^{M_1+k}(1 - z^{-1}.e^{-aT})}{1 - z^{-1}.2e^{-aT}\cos((\omega_\phi + 2kb)T) + z^{-2}.e^{-2aT}} +$$

$$\sum_{n=1}^{M_u} \frac{W_n(-1)^{M_1+N+n+1}(1 - z^{-1}.e^{-aT})}{1 - z^{-1}.2e^{-aT}\cos((\omega_{\phi p} + nb)T) + z^{-2}.e^{-2aT}} \qquad (3.3.1)$$

Now, in equation(3.3.1), the zeros have been moved close to the unit circle and made equal for all elemental filters. Since the zeros of all elemental filters are now made equal, the above equation can be written as

$$H(z) = (1-z^{-1}.e^{-aT})\left[\sum_{n=1}^{M_1} \frac{W_n(-1)^{M_1-n}}{1-z^{-1}.2e^{-aT}\cos((\omega_\phi - nb)T)+z^{-2}.e^{-2aT}} + \right.$$

$$\sum_{k=o}^{N-1} \frac{(-1)^{M_1+k}}{1-z^{-1}.2e^{-aT}\cos((\omega_\phi + 2kb)T)+z^{-2}.e^{-2aT}} +$$

$$\left.\sum_{n=1}^{M_u} \frac{W_n(-1)^{M_1+N+n+1}}{1-z^{-1}.2e^{-aT}\cos((\omega_{\phi p} + nb)T)+z^{-2}.e^{-2aT}}\right] \qquad (3.3.2)$$

We notice that by writing equation(3.3.1) in the form of equation(3.3.2), we have altered the physical realization of H(z). In equation(3.3.2), we have a nonrecursive section in cascade with a number of recursive second-order sections in parallel. This equation(3.3.2) now represents a further saving in hardware for the simple reason that we are now sharing the zeros among all elemental filters. However, this is by no means the end of the saving, if we further note that the zeros are now quite close to the unit circle for small value of a. This implies that we can virtually move the zeros onto the unit circle as represented below:

$$H(z) = (1-z^{-1})\left[\sum_{n=1}^{M_1} \frac{W_n(-1)^{M_1-n}}{1-z^{-1}.2e^{-aT}\cos((\omega_\phi - nb)T)+z^{-2}.e^{-2aT}} + \right.$$

$$\sum_{k=o}^{N-1} \frac{(-1)^{M_l+k}}{1-z^{-1}.2e^{-aT}\cos((\omega_+ +2kb)T)+z^{-2}.e^{-2aT}} + \sum_{n=1}^{M_u} \frac{W_n(-1)^{M_l+N+n+1}}{1-z^{-1}.2e^{-aT}\cos((\omega_+ +nb)T)+z^{-2}.e^{-2aT}} \Bigg\} \quad (3.3.3)$$

The term $(1-z^{-1})$ actually represents (x_n-x_{n-1}) in the difference equation for the general model which is clearly the differential input of the difference of the input signal x_n and its delayed version, x_{n-1}. Hence, in realizing the zeros of equation(3.3.3), we only need one subtraction and no multiplication at all, which is again a saving in hardware. This is very important in realtime digital filtering, where minimum multiplication time is the main objective, since multiplication is the essential factor governing both the throughput rate and the hardware volume required for a particular digital filter.

In this section we have discussed the zero-sharing property of the proposed technique for realtime filtering, in the next section, we shall describe another important aspect of this technique: pole-sharing property.

3.4 POLE-SHARING PROPERTY IN FILTER BANK IMPLEMENTATION:

As previously mentioned, in many realtime applications, our objective is to design a bank of n bandpass filters for frequency analysis purposes. A conventional approach would lead to a straightforward realization of n individual bandpass filters, though the frequency sampling design can offer a pole-sharing property which saves on the computational effort. On the other hand, a closer examination of our proposed technique would reveal an interesting pole-sharing property which also enables immense savings in the number of elemental filters used in the actual implementation of the filter bank.

If one were to use the proposed technique to design the filter bank in the above straightforward manner, one would require $n(M_l+M_u+N)$ elemental filters to realize the design. However, due

to the modularity of the resultant design which requires only the paralleling of a number of elemental filters to form a higher-order filter, some of the poles between adjacent bandpass filters are identical and can therefore be shared them. This further implies that only $x(M_l+M_u+N)$ additional elemental filters are needed for each bandpass filter, if x poles are shared between adjacent filters. For instance, if x=2, n=200, $(M_l+M_u+N)=4$, then a total of 200(4-2)+2 or approximately 400 poles are needed instead of 800 in the straightforward realization. Figure 3.4.1 depicts the situation in which adjacent filters, for example the nth and the (n-1)th filters, share their poles in their implementations.

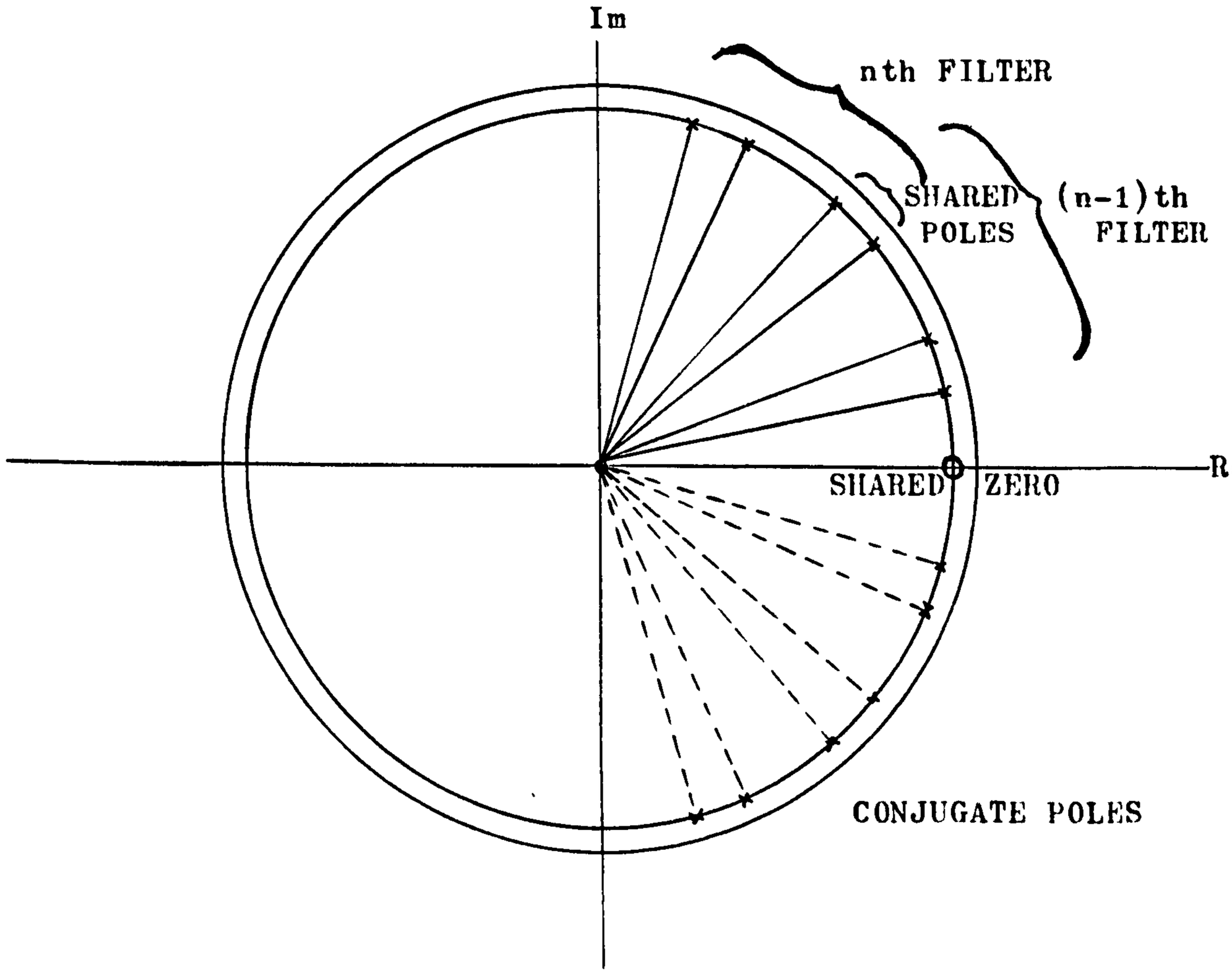
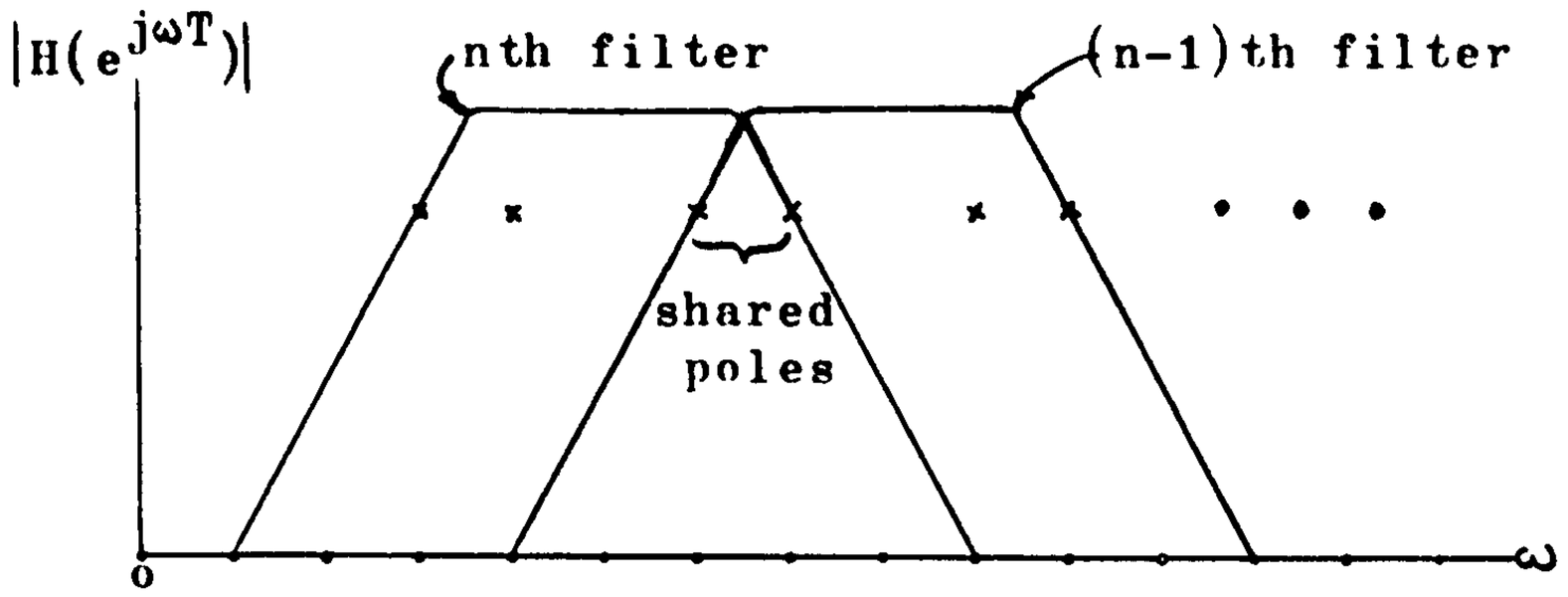


FIG: 3.4.1 POLE-SHARING STRUCTURE IN THE Z-PLANE. EACH COMPLEX POLE-PAIR OF AN ELEMENTAL FILTER IS REALIZED WITH A SECOND-ORDER DIGITAL RESONATOR.

In figure 3.4.1, the zero-sharing property is also shown in the z-plane. Each elemental filter is realized with a second-order digital resonator consisting of a complex pole-pair.

Figure 3.4.2 shows the response of the filter bank and the bandpass filters are each made up of two passband poles and two transition poles. In physical realization, single poles only exists in first-order digital filters with complex coefficients. Therefore, in our realization, we have chosen complex pole-pairs realized with second-order digital filters with real coefficients. Nevertheless, in the above illustration, we have referred to 4-pole bandpass filters in the filter bank for the sake of simplicity. These filters are realized with four elemental filters and the reader should also be aware of a conjugate 4-pole bandpass filter which exists for every real 4-pole bandpass filter in the bank.



SKETCH OF THE

FIG: 3.4.2 FREQUENCY RESPONSE OF THE POLE-SHARED FILTER BANK OF 4-POLE BANDPASS FILTERS.

In general, if the number of poles being shared is doubled, the magnitude of the savings available increases by unity. For instance, if $(M_l+M_u+N)=6$, x=4, then a total of 400 poles are needed in the above example of 200 filters in the bank, as opposed to 1200 poles in the straightforward realization, giving a saving of the order of magnitude 3.

3.5 COMMENTS ON CONVENTIONAL REALIZATION SCHEMES FOR THE PROPOSED DESIGN TECHNIQUE:

The resultant modularity of the proposed design yields a simple and modular approach to the realization of a digital filter. In this section, we shall examine the realizations of design equations (3.2.28), (3.3.1), (3.3.2) and (3.3.3) using conventional schemes. The details of these schemes used in this section are being given in section 4.1 in chapter 4.

Since higher-order filters simply means the paralleling of a number of elemental filters, equations (3.2.28) and (3.3.1) will have the following structure:

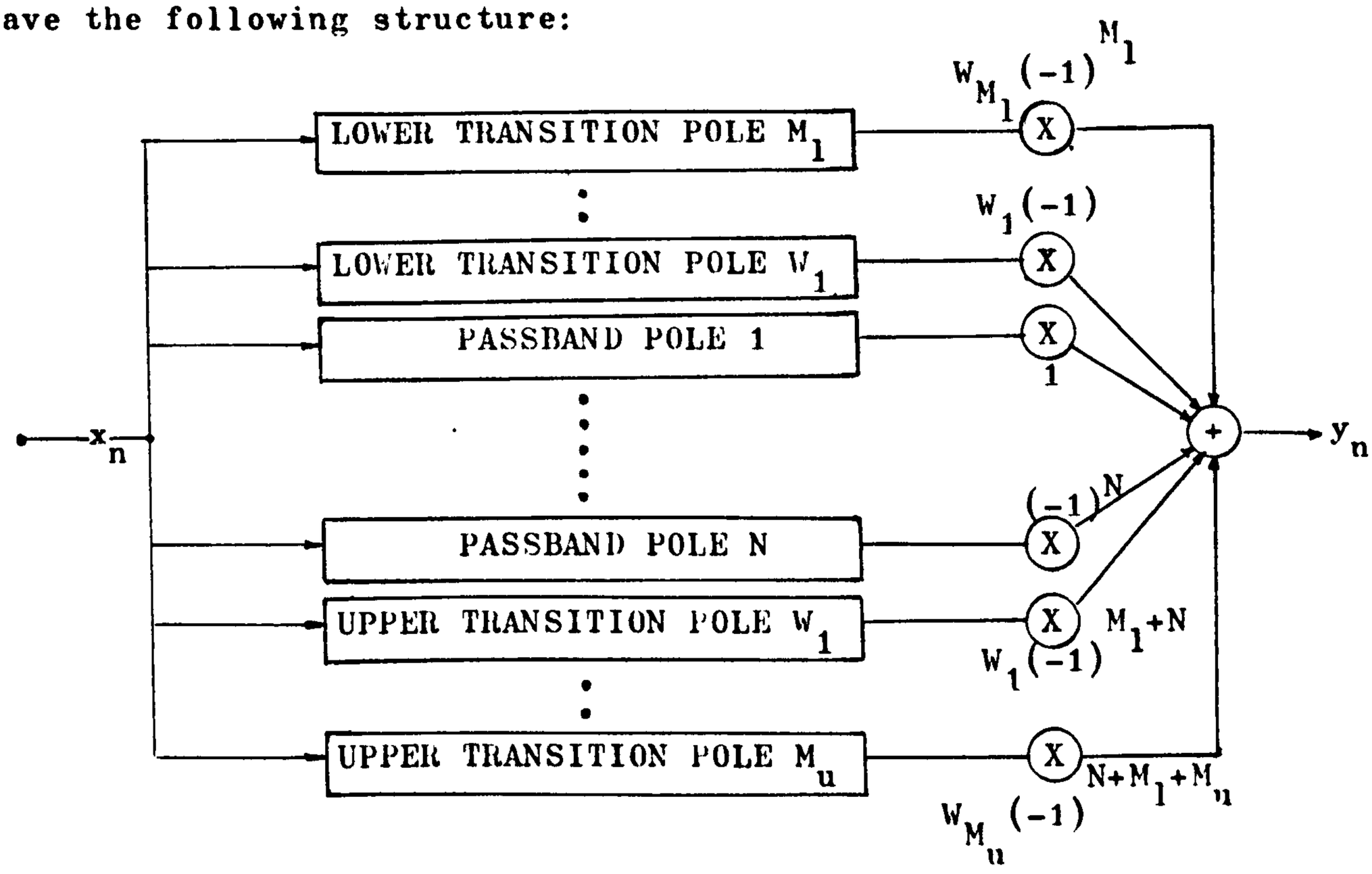


FIG: 3.5.1 PARALLEL REALIZATION SCHEME OF A BANDPASS DESIGN

Figure 3.5.1 shows a parallel realization scheme where all elemental filters are scaled with the optimized weightings and phase-modified. Their outputs are then summed to give the final output of the bandpass filter. Figure 3.5.2 shows the conventional scheme of realizing the poles structure.

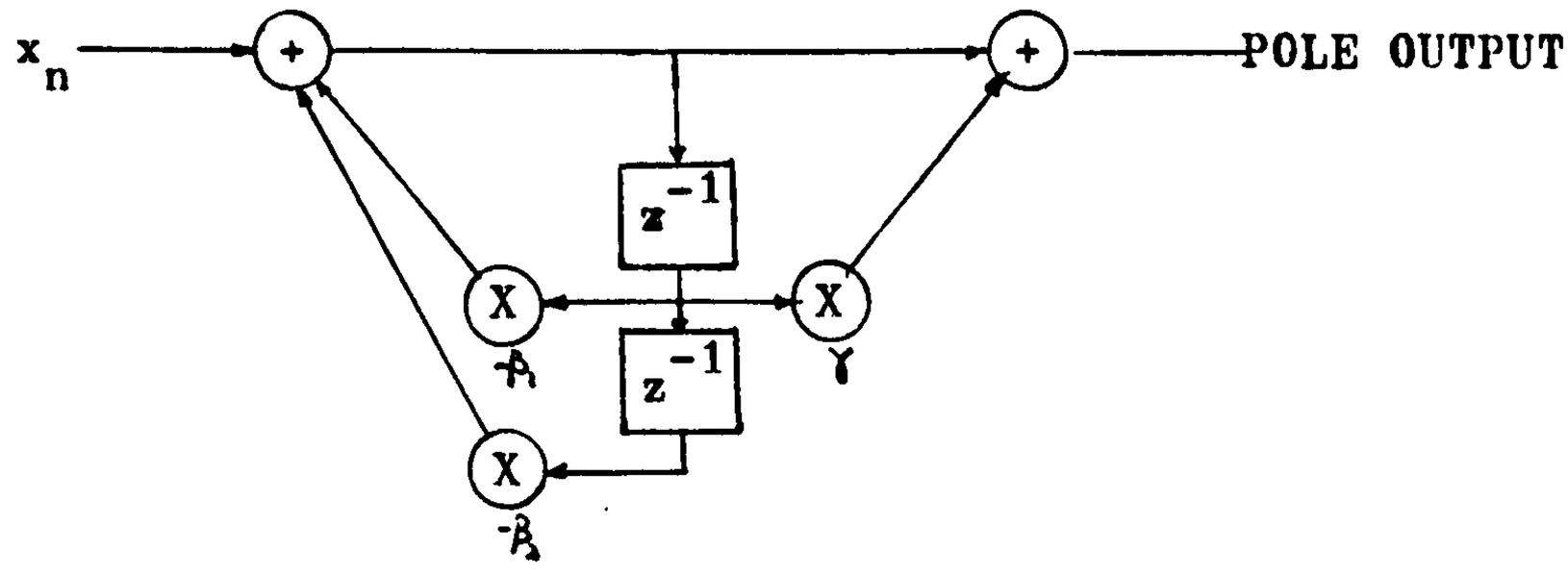


FIG: 3.5.2 CONVENTIONAL CANONIC STRUCTURE FOR AN ELEMENTAL FILTER AS A POLE

As mentioned before, the poles of the bandpass filter are the poles of the elemental filters used to approximate the desired bandpass response. Each pole of the bandpass filter can in fact be realized with a first-order section, however, in our implementation we use second-order sections instead. The use of second-order sections with real coefficients avoids the need of performing complex multiplications required if first-order sections were used instead. For equation(3.2.28), the coefficients of the elemental filters realizing the poles are:

$$\gamma = \begin{cases} -e^{-aT}\cos((\omega_p - nb)T) & \text{in the lower transition band,} \\ -e^{-aT}\cos((\omega_p + 2kb)T) & \text{in the passband,} \\ -e^{-aT}\cos((\omega_{tp} + nb)T) & \text{in the upper transition band.} \end{cases} \tag{3.5.1}$$

$$-\beta_1 = \begin{cases} 2.e^{-aT}\cos((\omega_p - nb)T) & \text{in the lower transition band,} \\ 2.e^{-aT}\cos((\omega_p + 2kb)T) & \text{in the passband,} \\ 2.e^{-aT}\cos((\omega_{tp} + nb)T) & \text{in the upper transition band.} \end{cases} \tag{3.5.2}$$

$$-\beta_2 = -e^{-2aT} \begin{cases} \text{in the lower transition band,} \\ \text{in the passband,} \\ \text{in the upper transition band.} \end{cases} \tag{3.5.3}$$

Similarly, we have for equation(3.3.1),

$$\gamma = -e^{-aT} \begin{cases} \text{in the lower transition band,} \\ \text{in the passband,} \\ \text{in the upper transition band.} \end{cases} \tag{3.5.4}$$

while the coefficients $-\beta_1$ and $-\beta_2$ are exactly the same as those in equations (3.5.2) and (3.5.3) respectively.

As previously mentioned in section 3.3, the multiplier γ in the above structure needs not be realized in practice for equation (3.2.28) for the second-order equations can be written as:

$$y_n = x_n - \beta_1(y_{n-1} - \tfrac{1}{2}x_{n-1}) - \beta_2 \cdot y_{n-2} \tag{3.5.5}$$

since $\gamma = \frac{1}{2}\beta_1$

Figure 3.5.2 can now be redrawn as follows:

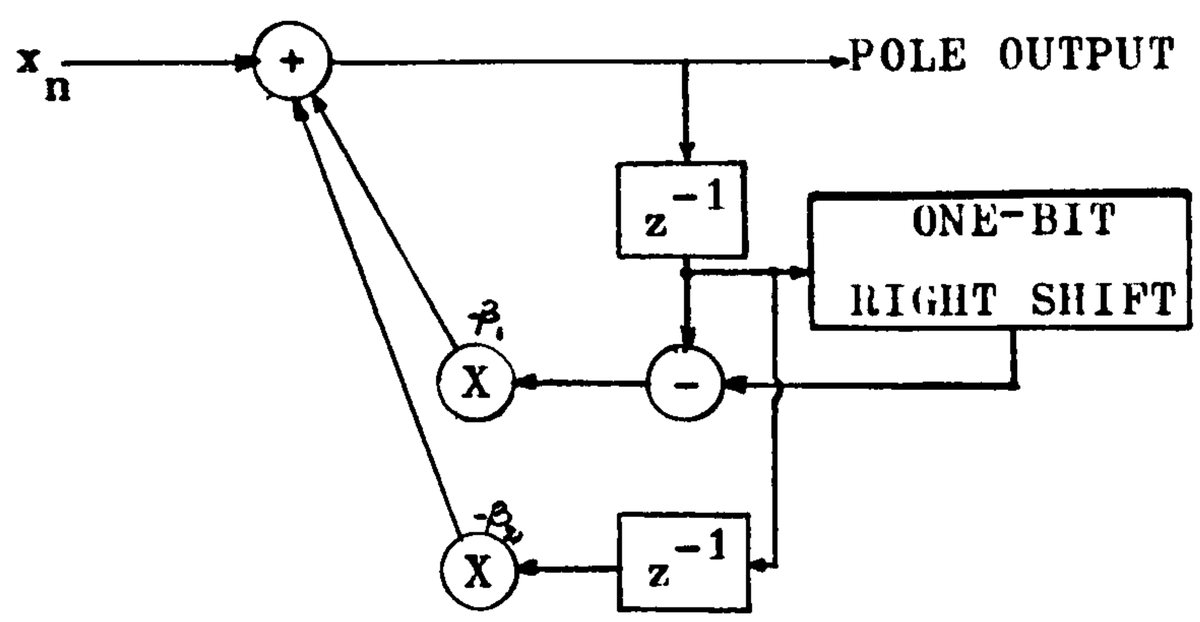


FIG: 3.5.3 ALTERNATIVE POLE REALIZATION SCHEME

Hence, only two multiplications are needed to yield an output sample from each elemental filter in equation(3.2.28) with the scheme in figure 3.5.3.

In the above, it is obvious that the pole structure is the same for all elemental filters, while they only differ in their filter coefficients. However, the realizations for equations (3.3.2) and (3.3.3) differ from those of equations(3.2.28) and (3.3.1) above slightly, in that the zeros of each elemental filter is being shared amount others as shown in figure 3.5.4.

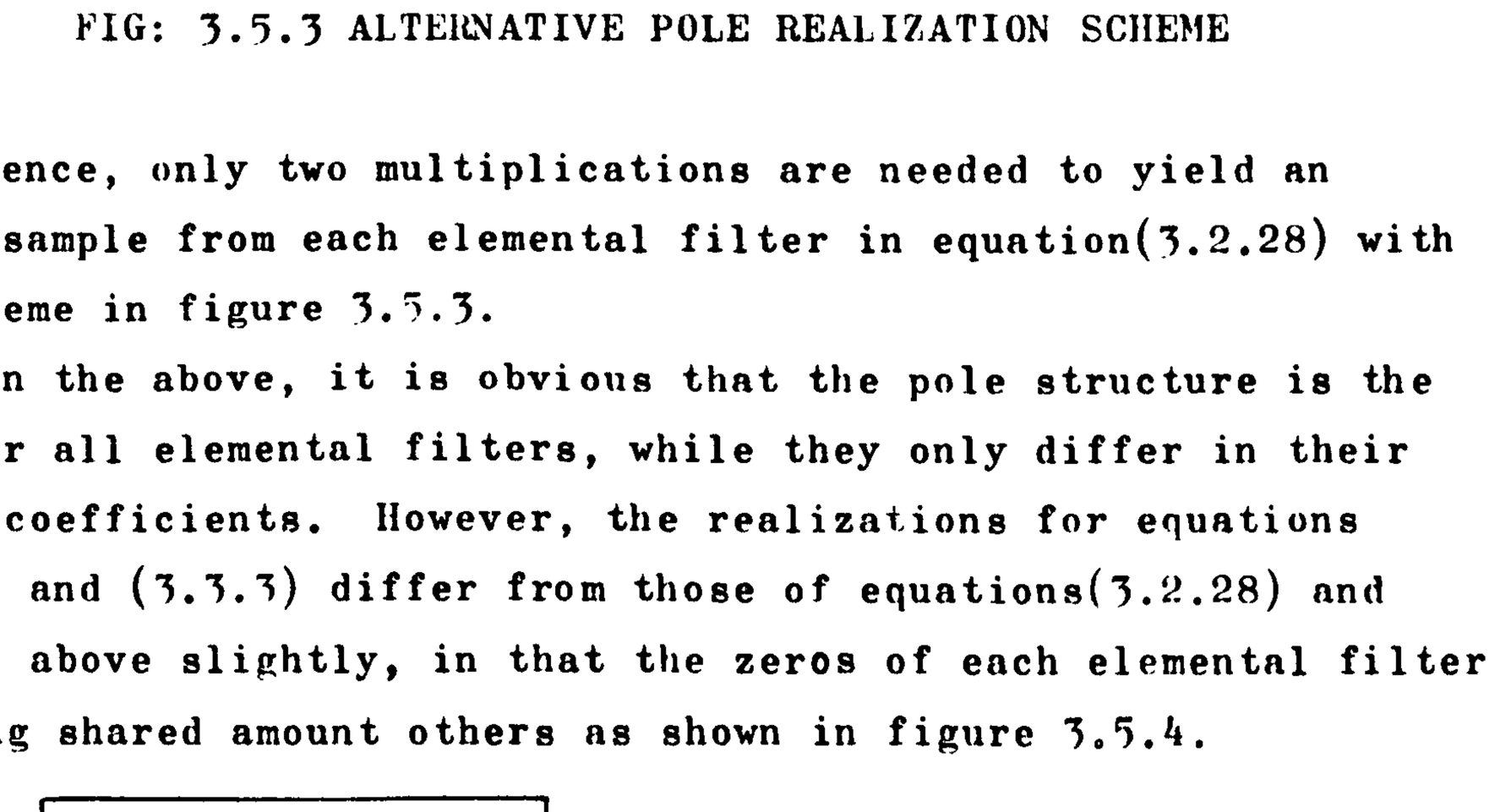


FIG: 3.5.4 ZERO SHARED REALIZATION OF A BANDPASS FILTER DESIGN

The scaling multipliers have the same coefficients as those shown in figure 3.5.1 where the outputs of individual elemental filters are optimized. The zero structure shown in figure 3.5.4 is a first-order nonrecursive section shared among all the inputs to the modified pole structures as depicted below:

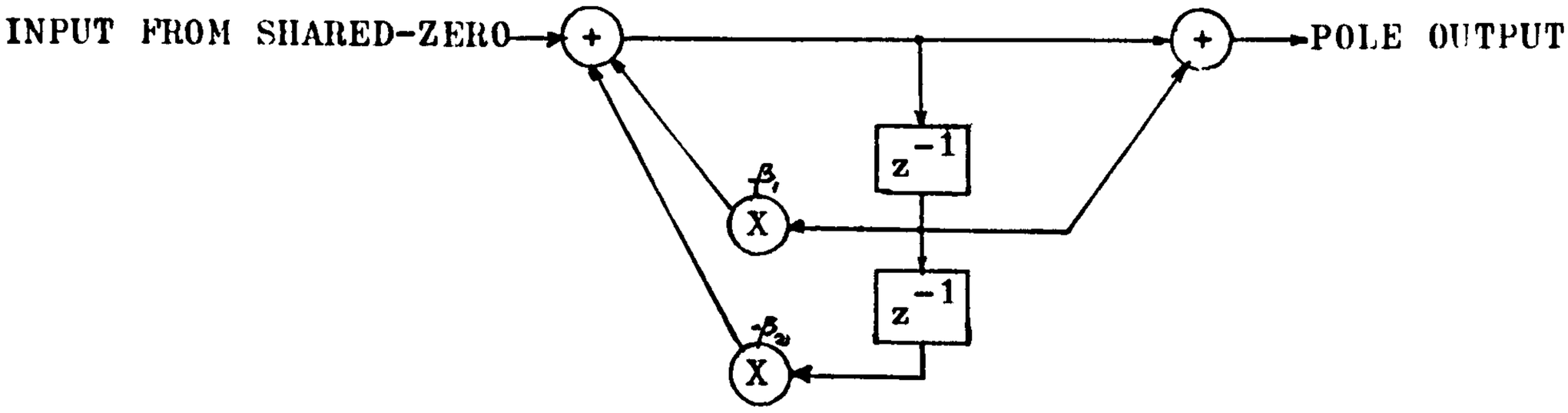


FIG: 3.5.5 POLE REALIZATION SCHEME FOR THE ZERO-SHARED STRUCTURE

The realization scheme for equation(3.3.3) will simply be that of figures 3.5.4 and 3.5.5, with the $-e^{-aT}$ multiplier in the nonrecursive shared-zero section in figure 3.5.4 made unity i.e. the input to the parallel combination of poles is now $x_n - x_{n-1}$.

Conceptually, the resultant transfer function from equations (3.2.28), (3.3.1), (3.3.2) and (3.3.3) can be represented as a vectorial combination of the elemental filters to form a composite frequency response as follows:

$$H(z) = H^T_{M_1}(z) - \ldots\ldots + (-1)^{M_1+1} H^T_1(z) + (-1)^{M_1+2} H^P_1(z) + \ldots\ldots + (-1)^{M_1+N+1} H^P_N(z) + (-1)^{M_1+N+2} H^T_1(z) + \ldots\ldots + (-1)^{M_1+M_u+N+1} H^T_{M_u}(z) \quad (3.5.6)$$

where $H^T(z)$ is an elemental filter response in the transition bands and $H^P(z)$ is an elemental filter response in the passband.

Finally, to conclude this section, we shall depict the realization scheme of the pole-shared structure of the filter bank in section 3.4 for n adjacent bandpass filters. Assuming that two transition and two passband poles are used for each bandpass filter, the configuration is shown in figure 3.5.6.

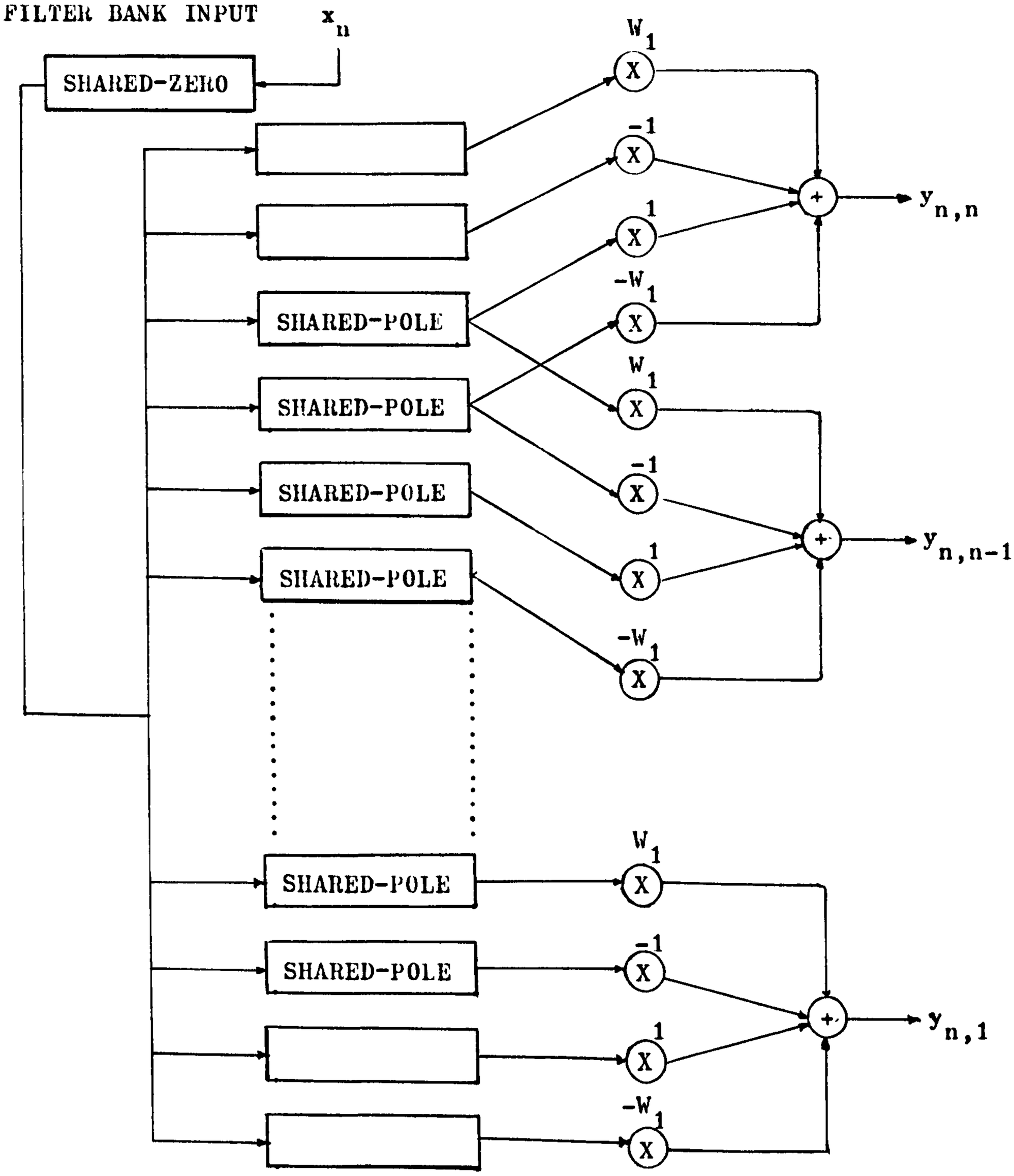


FIG: 3.5.6 REALIZATION SCHEME FOR THE ZERO-SHARED, POLE-SHARED FILTER BANK OF "n" BANDPASS FILTERS.

In figure 3.5.6, the outputs of the n bandpass filters are shown to be $y_{n,n}$, $y_{n,n-1}$, ... , $y_{n,1}$ respectively. In the above configuration, the passbands and transition bands of the n bandpass filters are made equal. In general, the passbands and transition bands of these filters need not be equal, and the number of passband poles and transition band poles can vary from filter to filter. As regards efficiency and savings, this section has pointed out the fact that the proposed design technique provides savings in terms of reductions in computational effort and hardware, even when being realized with conventional realization schemes. However, in chapter five, we shall discuss a novel hardware implementation of high-speed digital filters which provides more savings in terms of hardware, speed and power consumption. Nevertheless, the proposed design technique in approximating an arbitrary frequency response is highly modular in its implementation, and higher-order filters can readily be implemented by simply paralleling more elemental filters in the passband and/or the transition bands.

3.6 REALTIME CALCULATIONS OF FILTER COEFFICIENTS:

As has been mentioned earlier, realtime calculations of filter coefficients can cut down dramatically the coefficient storage requirement of a large number of filters, for example, in a realtime filter bank implementation. In this section, we point out an ~~interpolation~~ recursion formula for the realtime calculations of the coefficients of the elemental filters in equations(3.2.28), (3.3.1), (3.3.2) and (3.3.3). A careful examination of the above equations would reveal the fact that the coefficients of adjacent elemental filters differ only in the z^{-1} terms. Consider, therefore, any two adjacent elemental filters in equation(3.2.28) which can be conveniently represented as follows:

$$H_n(z) = \frac{(-1)^n W_n\left(1 - z^{-1}.e^{-aT}\cos(nbT)\right)}{1 - z^{-1}.2e^{-aT}\cos(nbT) + z^{-2}.e^{-2aT}} \qquad (3.6.1)$$

$$H_{n+1}(z) = \frac{(-1)^{n+1} W_{n+1}\left(1 - z^{-1}.e^{-aT}\cos(\overline{n+1}bT)\right)}{1 - z^{-1}.2e^{-aT}\cos(\overline{n+1}bT) + z^{-2}.e^{-2aT}} \qquad (3.6.2)$$

The cosine term in $H_{n+1}(z)$ can be obtained from that of $H_n(z)$ via the ~~interpolation~~ recursion formula

$$\cos(\overline{n+1}bT) = \cos(nbT+bT)$$
$$= \cos(nbT)\cos(bT)-\sin(nbT)\sin(bT) \qquad (3.6.3)$$

As the angle bT is small for large T and small b in narrowband filter implementation, therefore, cos(bT) varies between the values 1 and 0.9998 approximately i.e. $\cos(bT)\simeq 1$, and equation(3.6.3) can then be written as

$$\cos(\overline{n+1}bT) = \cos(nbT)-\sin(nbT)\sin(bT) \qquad (3.6.4)$$

Hence, $\cos(\overline{n+1}bT)$ can be represented approximately by the difference of cos(nbT) and the term sin(nbT)sin(bT) which represents the small interpolation value sin(bT) since the angle bT is small. Consequently, equation(3.6.4) represents an efficient way of calculating the coefficients of one elemental filter from another. This implies that the coefficients required in, say, a filter bank implementation, can be derived in realtime instead of prestored in a coefficient memory. For instance, in multiplexed digital filtering the coefficients of another channel can be calculated in realtime while the arithmetic unit of the digital filter is filtering its predecessor.

3.7 THE CHOICE OF SAMPLING FREQUENCY

If the elemental filters are derived from a continuous model, for example, by an Impulse Invariance transformation, it is true that if "quantization effects are ignored", the derived digital elemental filters approximate the continuous model closely as the sampling period T decreases. This is clear from the aliasing effects inherent in the Impulse Invariance transformation. However, when the bandwidth of individual poles is small compared with the sampling frequency, the above aliasing effects will be minimal. Thus, it is useful to define the index BT which is the ratio of the pole's bandwidth B to the sampling frequency 1/T.

The bandwidth of the poles of the elemental filters in equations(3.2.28), (3.3.1), (3.3.2) and (3.3.3) is given by

$$B=2a \qquad (3.7.1)$$

such that

$$BT=2aT \qquad (3.7.2)$$

This gives an indication of the "degree of aliasing" of the poles and hence their suitability in any design using the proposed approximation technique. However, in some cases, the invariance of the impulse response may be all that is required in the design, and this is by no means a function of the sampling frequency.

Alternatively, in the case of a bandpass filter design, the ratio of the passband or bandwidth of the filter to its centre frequency, termed the percentage bandwidth, gives also a useful indication of how close will the resultant design match the ideal specifications. For instance, in the case of narrowband designs, the percentage bandwidth will be very small and the resultant frequency response will suffer little aliasing effect if any at all.

In view of the above, our proposed approach therefore calls for high-Q narrowband elemental filters where the ratio BT becomes extremely small, and any aliasing effects present will be quite negligible. In addition, the proposed design technique is very suitable for narrowband filters, while wideband designs may require a large number of passband poles to yield a sharp response. As in above, the percentage bandwidth of a pole can be similarly defined as B/ω_r where ω_r is the resonant or centre frequency of the pole. This then serves as a useful index independent of the sampling frequency. Nevertheless, in some cases, an increase in the sampling frequency will help, but there is a certain limit when quantization effects of a finite wordlength processor is taken into account. These effects will be discussed in detail in chapter six.

3.8 CONCLUDING REMARKS:

In general, to meet a particular filter's specifications, the designer is often left with the options of either using a low-order design or an equivalent higher-order design. The choice appears to

be arbitrary, but in practice, depends very much on finite wordlength effects, such that the designer has to compromise for the best solution. Consider now an approximation of a frequency response H(z) where high-order poles are used instead of simple poles as in equation(3.2.28). Therefore, equation(3.2.28) will now be written as

$$H(z) = \sum_{n=1}^{M_1} W_n(-1)^n \left[\frac{1-z^{-1}.e^{-aT}\cos((\omega_p-nb)T)}{1-z^{-1}.2e^{-aT}\cos((\omega_p-nb)T)+z^{-2}.e^{-2aT}} \right]^P +$$

$$\sum_{k=1}^{N} (-1)^{M_1+k} \left[\frac{1-z^{-1}.e^{-aT}\cos((\omega_p+2kb)T)}{1-z^{-1}.2e^{-aT}\cos((\omega_p+2kb)T)+z^{-2}.e^{-2aT}} \right]^Q +$$

$$\sum_{n=1}^{M_u} W_n(-1)^{M_1+N+n} \left[\frac{1-z^{-1}.e^{-aT}\cos((\omega_{+p}+nb)T)}{1-z^{-1}.2e^{-aT}\cos((\omega_{+p}+nb)T)+z^{-2}.e^{-2aT}} \right]^P$$

where $P > 1$, $Q > 1$. (3.8.1)

Figure 3.8.1 depicts a possible realization scheme for equation (3.8.1) for $M_1=M_u=1$, N=2 and P=Q=2. The z-plane diagram will be the same as before except that we have all double poles instead of simple poles at each pole location.

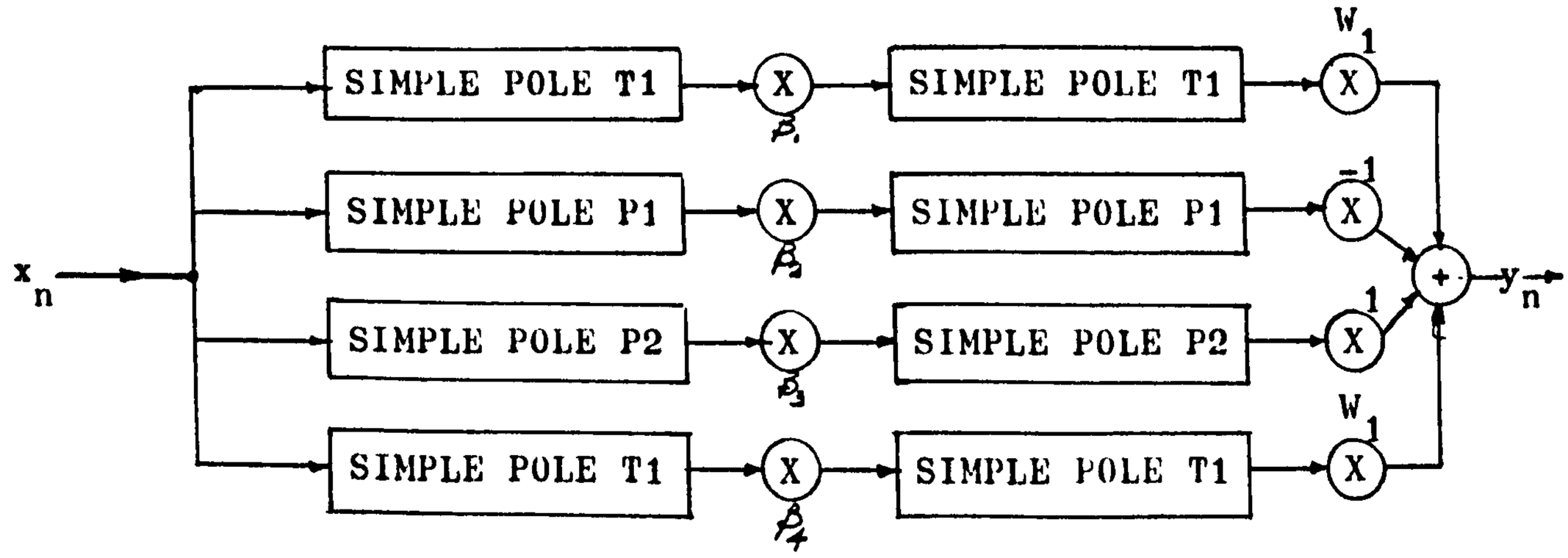


FIG: 3.8.1 A POSSIBLE HIGH-ORDER REALIZATION OF A 8-POLE BANDPASS FILTER

The scaling parameters β's are used to prevent filter overflow between stages.

In conclusion, we have proposed in this chapter a design technique, which is particularly suitable for designing filter banks, for high-speed realtime digital filtering. In this design technique, the use of a proposed frequency-sampling procedure results in a closed-form expression for a linear-phase equiripple passband structure which is further optimized by a proposed optimization procedure to yield a final optimum transfer function. Thus, the proposed design technique involves two steps. First, we sample, in frequency, the ideal spectrum of a digital filter to be approximated. The sampling is performed by distributing elemental filter responses, called the frequency samples, uniformly across the passband, so that the entire spectrum of interest is sampled "adequately". Next, we erect frequency samples in the transition band(s), called the transition samples, at closer equispaced intervals than those in the passband, to improve the transition characteristics and to provide increased flexibility in locating the cutoff frequencies in the passband and the stopband(s). The above elemental filters are high-Q resonators in the proposed design technique. In the second step of the proposed approximation of the desired response, we optimize the design to provide an optimum structure. In particular, the transition frequency samples are being optimized to yield the steepest transition skirt(s) possible. Under the proposed optimization procedure, the resultant or final filter response is tailored to suit individual specifications given, and in general, larger stopband attenuation and more reduced sidelobes can be obtained. Linear phase within the passband is maintained by having all poles at equal distance from the unit circle and also "phase-modified". Nevertheless, a novel feature of the above proposed design technique is the use of optimized high-Q resonators as elemental filters to sample in frequency along a circle within the unit circle, at two different sets of equispaced sampling points (one in the passband and the other in the transition band(s)), thereby producing a closed-form expression to approximate a desired response. In applying the proposed optimization procedure, more emphasis is placed on the constraints in the transition band(s) in such a way that the number of variables in the process have been reduced. Furthermore, the effects and significance of introducing these transition samples

in the proposed design technique have been thoroughly explored. An additional feature of the proposed design technique is the possibility of realtime calculations of the filter coefficients. Nevertheless, the proposed design technique offers a reduction in hardware, even when implemented with existing approaches in hardware implementation. This is feasible via the modularity of the resultant design which allows both zero- and pole-sharing as well as realtime computations of filter coefficients, so as to reduce the storage requirement of, for instance, a filter bank implementation. Such applications will be discussed in chapter nine in this thesis. Finally, the modularity feature of the proposed design technique means high-order filters can be achieved simply by parallelling more lower-order (sub) filters (or elemental filters).

CHAPTER FOUR

A REVIEW OF CONTEMPORARY HARDWARE IMPLEMENTATIONS OF DIGITAL FILTERS

4.1 INTRODUCTION: (47)

A digital filter can be implemented either by programming a general-purpose computer, as a computer routine, or by constructing a special-purpose digital processor using electronic hardware. In section 3.5, we outlined and commented on the conventional approach to the realization of the proposed design technique for realtime filtering. In this chapter, we shall review contemporary hardware implementations of these realization schemes.

In general, one almost always realizes an Nth order digital filter as a cascade or parallel combination of first-order and second-order digital filters in order to assume that small errors in the coefficients, due to rounding, do not result in unacceptably large errors in pole positions(44) which may further lead to instability. The parallel form of an Nth order digital filter realization results from a partial fraction expansion of the given transfer function in its direct form

$$H_d(z) = \frac{\sum_{j=0}^{N} a_j . z^{-j}}{1 + \sum_{j=1}^{N} b_j . z^{-j}} \qquad (4.1.1)$$

into the form

$$H_p(z) = \gamma_o + \sum_{j=1}^{M} \frac{\gamma_{1j} . z^{-1} + \gamma_{oj}}{\beta_{2j} . z^{-2} + \beta_{1j} . z^{-1} + 1} \qquad (4.1.2)$$

while the cascade form corresponds to a factorization of the numerator and denominator polynomials of equation(4.1.1) to bring forth the form

$$H_c(z) = a_o \prod_{j=1}^{M} \left(\frac{\alpha_{2j} . z^{-2} + \alpha_{1j} . z^{-1} + 1}{\beta_{2j} . z^{-2} + \beta_{1j} . z^{-1} + 1} \right) \qquad (4.1.3)$$

where M is the least integer greater than or equal to N/2.

The digital circuit layouts corresponding to equations(4.1.2) and (4.1.3) are shown in figures (4.1.1) and (4.1.2) respectively.

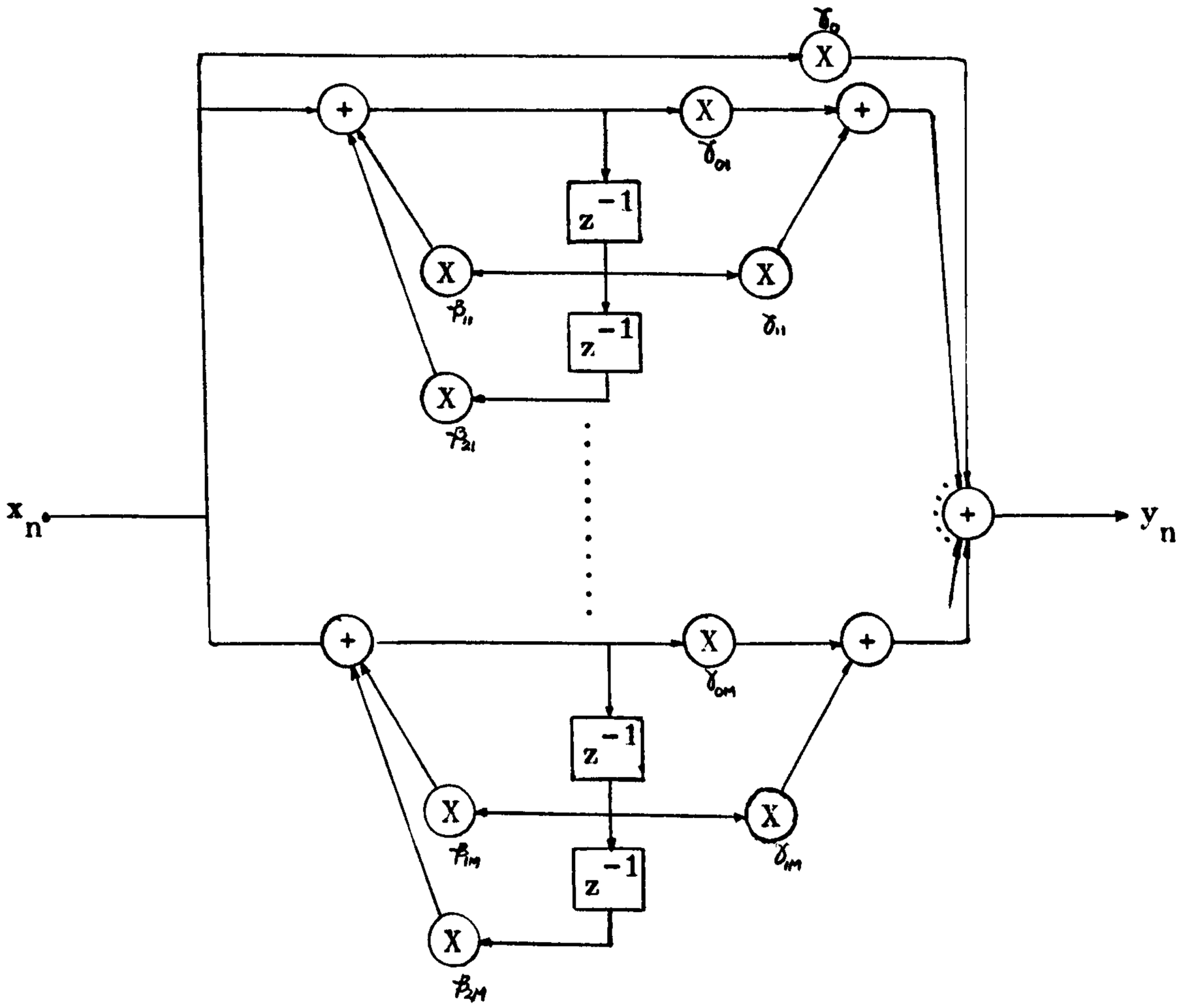


FIG: 4.1.1 THE PARALLEL FORM OF REALIZATION OF A GENERAL Nth ORDER DIGITAL FILTER

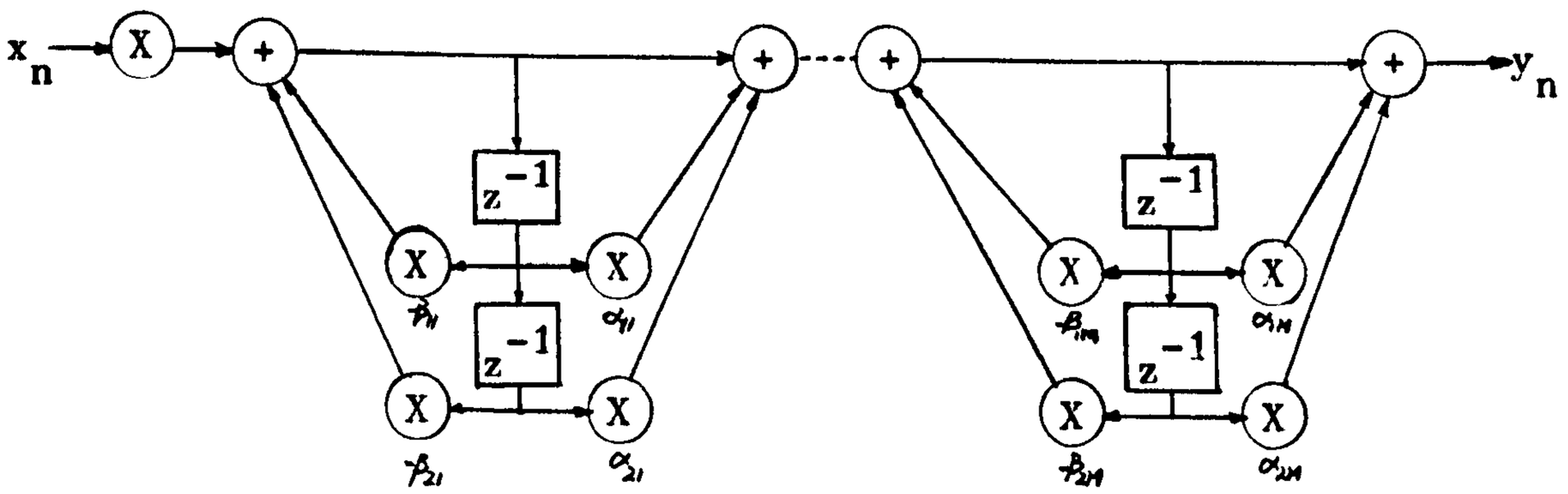


FIG: 4.1.2 THE CASCADE FORM OF REALIZATION OF A GENERAL Nth ORDER DIGITAL FILTER

4.2 HARDWARE IMPLEMENTATIONS USING CONVENTIONAL MULTIPLIERS AND ADDERS:

In this section, we first distinguish the notions behind multiplications in digital filter circuits by conventional multipliers and by memory technique. In a conventional multiplier, the partial products of the multiplication process are formed by some logic gates, while their summation to form the final product is performed by conventional adders. Consequently, the speed of the multiplication process depends on the total logic gate-delay in the partial products formation and the total addition time spent on summing these partial products to form the final products.

In contrast, when the equivalent function of a conventional multiplier is performed by the use of memory, the partial products are precalculated and stored in a memory while being read out later to form the final product. This memory technique is often referred to as table lookup, and the summation of the partial products in this multiplication process is performed by conventional adders as well. The speed of multiplication via the memory technique is thus the memory access time plus the partial products summation time. In general, the partial products formed in the above two multiplication processes are different and hence their summation times are also different. Nevertheless, a conventional adder forms the sum of two numbers by a connection of some logic gates while the above memory technique can also be extended to perform its equivalent function without any logic functions.

In section 4.1, figures 4.1.1 and 4.1.2 show the essential digital components for the hardware implementation of a digital filter, viz., multipliers, adders and delay elements. The multiplications of the filter variables by the constant filter coefficients are performed by conventional multipliers unless superseded by memory techniques which will be explained in the next section. The delay elements are realized with shift registers and addition performed by conventional adders or memory techniques. Furthermore, all filter coefficients are prestored in a seperate coefficient memory.

The paper by Jackson, Kaiser and McDonald (45) can be regarded as a classic in the hardware implementation of digital filters using

conventional multipliers. In their approach, they have used a serial-mode multiplier for the circuit realizations of figures 4.1.1 and 4.1.2. Their serial-mode multiplier multiplies two numbers in a bit-by-bit manner while the summation of partial products is also in a serial manner. The advantage of their serial implementation is economy in hardware and a tradeoff between accuracy and processing speed of the resultant digital filter. However, in general, a serial-mode implementation is much slower in comparison with a parallel-mode implementation and the operational speed depends largely on the wordlength of the processor.

On the other hand, the paper by Gabel (46) describes a parallel implementation in contrast to the above serial implementation. Unlike serial-mode multipliers which multiply one bit at a time, the parallel-mode multipliers are theoretically independent of the wordlength used. While an increase in internal wordlength of a serial implementation necessitates a longer processing time, the parallel implementation treats all bits of filter variables and coefficients simultaneously, thus requiring a minimum processing time. Hence, in the parallel mode, the tradeoff is between hardware and accuracy as considerably more hardware is used when compared with the serial mode. Furthermore, in a serial implementation, the serial multiplier yields partial products individually and their summation is performed two at a time, thus the complexity of the control circuitry increases. On the other hand, all partial products are formed simultaneously by a parallel multiplier and summed all at once, therefore, the control circuitry required is a minimum.

In short, multipliers are in essence a major factor in determining the operational speed, power consumption and hardware complexity of the implementations of digital filters. As such, the ultimate cost of digital filters implemented with conventional multipliers will depend very much on the "efficiency" in employing them. Mori, in his paper (48), has proposed a streamlined operation of digital filters using conventional multipliers in a "cellular structure" of their arithmetic units. Besides array multipliers (10) (49),(50),(51),(52), pipeline multipliers (53),(54),(55) have also been used to speed up the filtering process by the efficient use of multipliers. On the other hand, Bin-Nun and Woodward have proposed

a modular approach (56) in the hardware implementation of digital filters which greatly improved the use of conventional multipliers. Moreover, many important algorithms (57),(58),(59), and designs of high-speed conventional multipliers and adders (60)-(68) have been proposed, however, their speeds have been mainly limited by the fundamental carry propagation path (69) in all digital systems.

In conclusion, it might be worth pointing out that most hardware implementations of digital filters rely mainly on the above efficient uses of multipliers and adders. For high-speed operation, the parallel mode is predominant, necessitating the use of high-speed multipliers and adders which often increases the hardware required and power consumption. In particular, array multipliers are often used in conjunction with carry-save and/or carry-look-ahead adders for speed, but, at the expense of excessive hardware and power consumption, let alone the cost of implementation. Other contemparary hardware implementations include memory technique which is the topic of the next section.

4.3 REPLACEMENT OF MULTIPLIERS BY MEMORY TECHNIQUE:

The use of semiconductor memories as multipliers in digital filters dates back to as far as the late 1960's (70). In his paper (71), Hemel shows a straightforward realization of multipiers using read-only-memories (ROM's) and conventional adders. However, he also points out that the straightforward approach will be as good as any other method but for the fact that if the wordlengths of the numbers to be multiplied increase beyond four bits, then the size of the memory required grows very rapidly. For instance, for two 5-bit numbers, the total word count is $2^{(5+5)}=1024$, and the output must be 10-bit long, that is a total bit count of 10240. For two 6-bit numbers, the total bit count is 49152, while for two 8-bit numbers, the situation has become ~~a bit~~ impractical for the bit count is now more than a million bits. No single memory with a million bits is available at present, and if a number of smaller memories are interconnected to achieve this size, the cost becomes prohibitive let alone the speed of the multiplier. To overcome this problem, some efficient algorithms have been used which will be introduced in the coming sections.

4.3.1 BYTE-SLICED TECNIQUE IN ROM MULTIPLICATION:

By using a more subtle means to produce an equivalent table lookup multiplier, the bit count of the table lookup memory has been reduced to 8192 for two 8-bit numbers. The algorithm used to achieve this reduction in bit count is based on the fact that one can represent each of the numbers to be multiplied, A_8 and B_8, as the sum of two smaller bytes. One byte is simply the last four bits of the given numbers and the other is the first four bits followed by four zeros. Thus,

$$A_8 = A_4 + \Delta A_4 \qquad (4.3.1)$$

where $$A_4 = XXXX0000 \qquad (4.3.2)$$

and $$\Delta A_4 = \quad XXXX \qquad (4.3.3)$$

and each X represents a binary 0 or 1. The other factor B_8 is similarly subdivided. Hence, with these representations, one can work out an algorithm for the original multiplication which can now be represented and reconstructed as the sum of several 4-bit multiplications:

$$A_8 . B_8 = (A_4 + \Delta A_4)(B_4 + \Delta B_4)$$

$$= A_4 . B_4 + A_4 . \Delta B_4 + \Delta B_4 . B_4 + \Delta A_4 . \Delta B_4 \qquad (4.3.4)$$

These four products and their sum can be implemented in four 2048-bit ROM's and five 4-bit adders as shown in figure 4.3.1. The multiplication time of the system is equal to the ROM access time plus the addition time of two layers of adders. Hence, if the memory size is comparatively large, then its access time will be longer correspondingly, thus lengthening the multiplication process.

We next consider a general form of the above algorithm for two unsigned binary numbers, while signed numbers require a modification of the existing algorithm. Let the two numbers be H(p bits) and K (q bits) as follows:

$$H = h_p \ldots\ldots\ldots h_3 h_2 h_1 h_o \qquad (4.3.5)$$

$$K = k_q \ldots\ldots\ldots k_3 k_2 k_1 k_o \qquad (4.3.6)$$

where h_o and k_o are the least significant bits and h_p and k_q are the most significant bits. H may be expressed as

$$H = \ldots\ldots\ldots C + B + A \ldots\ldots\ldots \qquad (4.3.7)$$

where
$$A = \ldots\ldots 000 h_{n-1} \ldots\ldots h_3 h_2 h_1 h_o \qquad (4.3.8)$$

$$B = \ldots\ldots 000 h_{2n-1} \ldots\ldots h_n 000 \ldots \qquad (4.3.9)$$

$$C = \ldots\ldots 000 h_{3n-1} \ldots\ldots h_{2n} 000 \ldots \qquad (4.3.10)$$

and n is an integer greater than unity and less than the total number of bits in H. Similarly, K may be expressed as

$$K = \ldots\ldots\ldots R + Q + P \ldots\ldots\ldots \qquad (4.3.11)$$

where
$$P = \ldots\ldots 000 k_{m-1} \ldots\ldots k_3 k_2 k_1 k_o \qquad (4.3.12)$$

$$Q = \ldots\ldots 000 k_{2m-1} \ldots\ldots k_m 000 \ldots \qquad (4.3.13)$$

$$R = \ldots\ldots 000 k_{3m-1} \ldots\ldots k_{2m} 000 \ldots \qquad (4.3.14)$$

and m is an integer greater than unity and less than the total number of bits in K.

It follows from equations (4.3.5) to (4.3.14) that the product HK can be expressed as

$$\begin{aligned} HK = {} & AP + AQ + AR + \ldots\ldots\ldots + \\ & BP + BQ + BR + \ldots\ldots\ldots + \\ & CP + CQ + CR + \ldots\ldots\ldots\ldots \end{aligned} \qquad (4.3.15)$$

Now, each of the "intermediate products" or partial products such as AP may be derived from a ROM of $2^{n+m}(n+m)$ bits capacity. These "intermediate products" may then be added to form the product HK.

Addition of the "intermediate products" is simple to implement. Consider the product BQ: it is the product of an n-bit byte and an m-bit byte and can be simply written as

$$BQ = (000....000h_{2n-1}.....h_n000..)(000...000k_{2m-1}..k_m00..)$$

$$(h_{2n-1}....h_n)(k_{2m-1}.....k_m).2^{m+n} \qquad (4.3.16)$$

It will thus be apparent that the "intermediate products" must, in general, be weighted by a factor 2^{jn+lm}, where j and l being integers depending on the significance of the multiplicants of the " intermediate product". Since all "intermediate products" consist of (n+m) bits, it follows that the addition of certain of these products, for example, (AP +BQ) may be carried out without any adder.

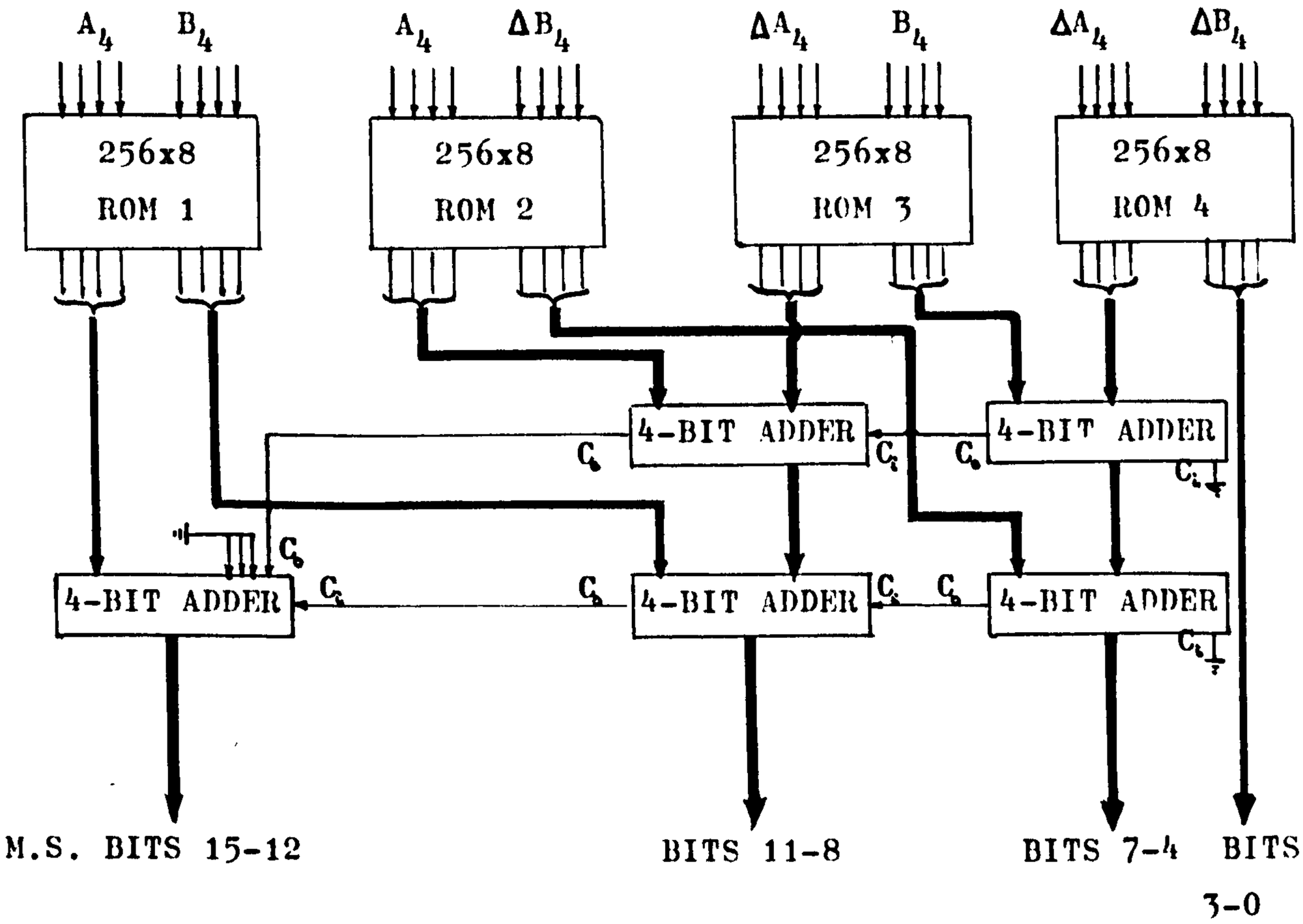


FIG: 4.3.1 ROM MULTIPLICATION OF TWO 8-BIT NUMBERS

However, despite a tremendous reduction in memory requirement when compared with the direct implementation, this algorithm also needs a rapidly increasing memory size when the wordlengths of the multiplicants are long i.e. large p and q. On the other hand, when n and m are small, the number of adders required rapidly increases and so is the number of adder layers, thus lengthening the

corresponding multiplication time. In general, this algorithm works well with sign-and-magnitude representation of signed binary numbers which requires the minimum amount of modification. However, when two's complement representation is used, the weighting factor in equation(4.3.16) is no longer so simple. Consequently the length of the modified "intermediate products" in this case varies according to the rule of signs in the two's complement notation(72).

In short, the algorithm works on a higher radix representation of the two numbers to be multiplied (56), which is naturally more efficient than radix 2. This further implies byte-slicing the two numbers into bytes shorter than their original wordlengths. The decomposition of numbers into bytes for multiplication is known as the "byte-sliced" technique.

B.Liu et.al.(73) have considered the application of the above algorithm to the implementation of digital filters and obtained a further reduction by noting the fact that the filters coefficients are constants. However, they have also pointed out that when signed numbers are involved, then the multiplication process will not be so straightforward. For instance, if two's complement representation is used, one has to differentiate between the four possible cases which the signs of the multiplicants can have, viz., (+,+),(+,-),(-,+)(-,-). This then involves the use of different ROM's for different cases, thus complicating the whole process. If one is to use sign-and-magnitude notation, one still has to inspect the signs of the numbers to be multiplied for the above four different cases, in order to decide the sign of their product. However, when sign-and-magnitude notation is being used, the inherent benefits of using the two's complement representation are lost. These include the advantages that addition of numbers are performed irrespective of their signs and intermediate sums and/or differences overflows are self-corrected provided that the final sum/difference is in-range.

4.3.2 BIT-SLICED TECHNIQUE IN ROM MULTIPLICATION:

B.liu et.al. (74)-(77) have also made use of a "bit-sliced" technique to implement a general Nth order difference equation of the form:

$$y_n = \sum_{k=0}^{N} a_k \cdot x_{n-k} - \sum_{k=1}^{N} b_k \cdot y_{n-k} \qquad (4.3.17)$$

while also noting the fact that the coefficients $\{a_k\}$ and $\{b_k\}$ are constants. In contrast to the "byte-sliced" approach, this method computes the sum of products of filter coefficients and "bit-sliced" filter variables by accumulating precomputed results stored in a ROM. In so doing, a string expression containing a summation of 2N+1 multiplications can be evaluated simultaneously by storing the above partial products in a single ROM instead of having individual ROM's for different filter coefficients as in the "byte-sliced" technique. This approach has thus led to significant improvement in speed and savings in hardware and power consumption. In fact, its economy has been compared with the "byte-sliced" technique (73) and can be measured by the (time)x (hardware) product which can have the unit of power consumption.

B.Liu et.al. have since then called filters designed by the "bit-sliced" technique "combinatorial digital filters", and by far this approach is to-date the most economical one in terms of speed and hardware (74),(75). The importance of the "bit-sliced" technique has warrented an investigation into its application in the hardware implementation of digital filters. Consider a basic second-order section below:

$$y_n = a_o \cdot x_n + a_1 \cdot x_{n-1} + a_2 \cdot x_{n-2} - b_1 \cdot y_{n-1} - b_2 \cdot y_{n-2} \qquad (4.3.18)$$

with $a_2=0$ in the case of parallel realizations. Assuming that all signals are bounded by $\pm$ 1 and that B binary bits including sign in two's complement notation are used to represent the data, that is,

$$x_n = -x_k^o + \sum_{j=1}^{B-1} x_k^j \cdot 2^{-j} \qquad x_k^j \in \{0,\ 1\} \qquad (4.3.19)$$

where $-x_k^o$ is the weighted sign bit.

Substituting equation(4.3.19) in (4.3.18), we have,

$$y_n = a_o\left(\sum_{j=1}^{B-1} x_n^j.2^{-j} - x_n^o\right) + a_1\left(\sum_{j=1}^{B-1} x_{n-1}^j.2^{-j} - x_{n-1}^o\right) +$$

$$a_2\left(\sum_{j=1}^{B-1} x_{n-2}^j.2^{-j} - x_{n-2}^o\right) - b_1\left(\sum_{j=1}^{B-1} y_{n-1}^j.2^{-j} - y_{n-1}^o\right) -$$

$$b_2\left(\sum_{j=1}^{B-1} y_{n-2}^j.2^{-j} - y_{n-2}^o\right) \qquad (4.3.20)$$

Now, B.Liu et.al. define a "coefficient-slicing" (74),(75) function ϕ with five binary arguments as follows:

$$\phi(x^1,x^2,x^3,x^4,x^5) = a_o.x^1 + a_1.x^2 + a_2.x^3 - b_1.x^4 - b_2.x^5 \qquad (4.3.21)$$

then putting (4.3.21) in (4.3.20), we have,

$$y_n = \sum_{j=1}^{B-1} 2^{-j}.\phi(x_n^j,x_{n-1}^j,x_{n-2}^j,y_{n-1}^j,y_{n-2}^j) -$$

$$\phi(x_n^o,x_{n-1}^o,x_{n-2}^o,y_{n-1}^o,y_{n-2}^o) \qquad (4.3.22)$$

While the function ϕ can be realized by a combinatorial logic network of 32 distinct states (hence the name combinatorial filter) or a ROM with 32 locations, addressed by the five arguments of ϕ, the decomposition of numbers into their binary bits for multiplication is known as the "bit-sliced" technique.

Now, equation(4.3.22) suggests a possible mechanization of equation(4.3.18), requiring only additions and shift-operations but without any multiplications involved. B.Liu et.al. suggest a hardware configuration as depicted in figure 4.3.2 for a second-order section of an Nth order digital filter. The operation of this filter as described by B.Liu et.al. is as follows:

Data enter serially into delay elements z_1^{-1} to z_4^{-1} with the least significant bit leading. At each shift, a new vector $\phi^j=\{x_n^j,x_{n-1}^j,x_{n-2}^j,y_{n-1}^j,y_{n-2}^j\}$ appears at the input of the circuit realizing ϕ. The output ϕ is loaded into register R_1 which is

connected to one of the two inputs of the accumulator with a possible sign change for j=o. The other input of the accumulator is hardwired to the output register R_2 with a 1-bit right shift. After B such shifts, the value in register R_2 is rounded and the accumulator cleared. This rounded value is y_n, which is shifted serially into z_4^{-1}, and the processor is ready to compute the next output sample y_{n+1}.

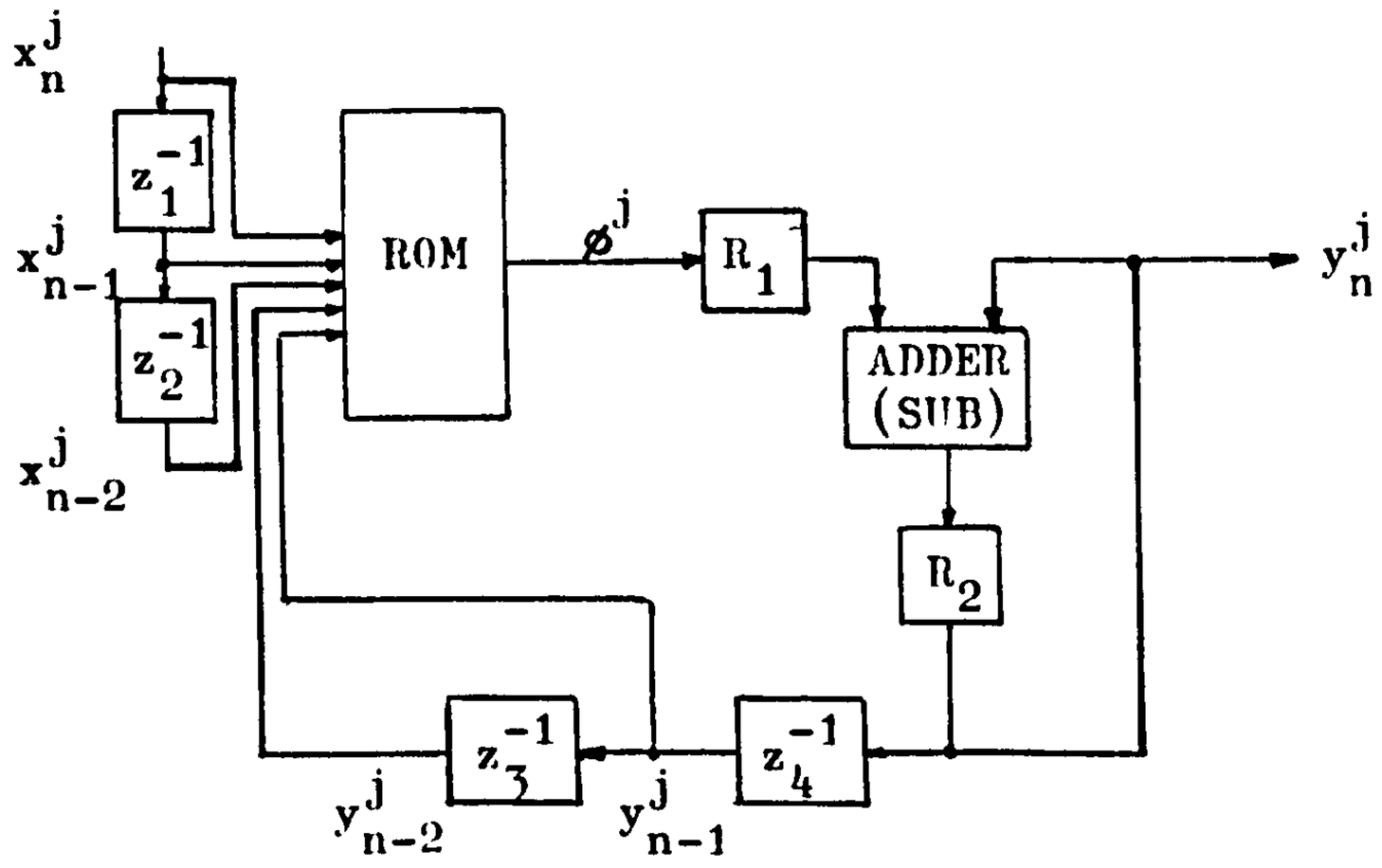


FIG: 4.3.2 SERIAL MECHANIZATION SCHEME OF THE "BIT-SLICED TECHNIQUE"

However, some comments are now in order, regarding the practical implementation of this filter section. To begin with, B.LIU et.al. are using two's complement notation so that the sign bit always carries a negative weight, thus contradicting their statement: "with a possible sign change for j=o". Moreover, due to the 1-bit hardwired right shift, modifications are necessary to compensate the weighting of the partial products and to prevent "end-overflows" during each add-and-shift operation. If rounding is done at the end of the multiplication process, then there is a definite loss in processing time of at least one clock cycle. Furthermore, if the rounded product is shifted serially into the z_4^{-1} shift-register according to B.Liu et.al., a further delay in time of B clock pulses is inevitable. Assuming the best case in

this serial mechanization, y_n will be produced only after a minimum delay of (B+1) clock cycles. Moreover, Wang (78) has pointed out that in B.Liu et.al. proposal, the operational speed (word rate) of Infinite-Duration-Impulse-Response (IIR) filters is different from that of the Finite-Duration-Impulse-Response (FIR) filters; and in their parallel mechanization (shown in figure 4.3.3), the difference is even larger.

In the serial mechanization above, the minimum clock period is given by

$$t_c = \max\{t_m,\ t_a\} \tag{4.3.23}$$

where t_m = ROM access time,
and t_a = addition time of the accumulator.

In their parallel mechanization, B.Liu et.al. have called for a configuration using B ROM's for the variables and a Wallace adder tree (65) to sum all the partial products. All of the B ROM's are simultaneously addressed and the partial products thus derived are added together according to their respective weightings. Again, if two's complement notation is used, modifications have to be made to readjust the various weightings of the partial products as mentioned before. By providing a pair of registers for each adder in figure 4.3.3, the operating clock period will be equal to the larger of t_m and t_α, i.e.

$$t_c = \max\{t_m,\ t_\alpha\} \tag{4.3.24}$$

where t_m = memory access time,
and t_α = addition time of one layer of adders.

However, assuming the best case, the product y_n always appears after $t_m(1+\alpha)$ or $t_\alpha(1+\alpha)$ depending on t_α or t_m is greater and α is the number of layers of adders used. Wang has also shown that in general, the word rate for the high-speed or parallel mechanization with two additional registers for each adder is given by

$$\frac{1}{(1+\log_2 B)\,t_c} \tag{4.3.25}$$

for $\log_2 B$ an integer.

For non-integral value of $\log_2 B$, we have,

$$\frac{1}{(1+\lceil 1+\log_2 B\rceil)\, t_c} \qquad (4.3.26)$$

where $\lceil x \rceil$ means the smallest integer of x.

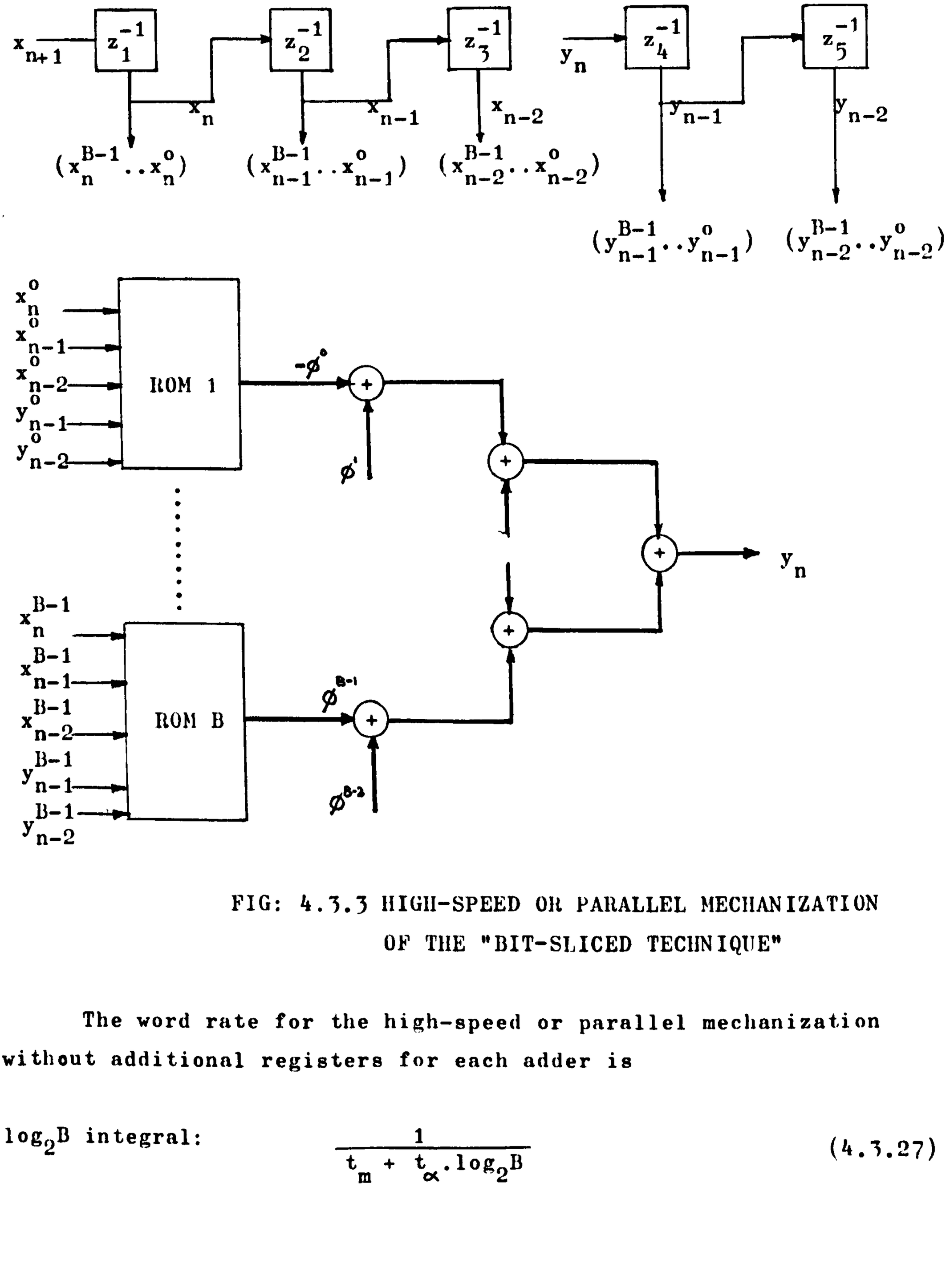


FIG: 4.3.3 HIGH-SPEED OR PARALLEL MECHANIZATION OF THE "BIT-SLICED TECHNIQUE"

The word rate for the high-speed or parallel mechanization without additional registers for each adder is

$\log_2 B$ integral:

$$\frac{1}{t_m + t_\alpha . \log_2 B} \qquad (4.3.27)$$

$\log_2 B$ non-integral: $$\frac{1}{t_m + t_\alpha\left[1 + \log_2 B\right]} \qquad (4.3.28)$$

Finally, if the additional registers for each adder are used in the FIR nonrecursive digital filter case, no additional time delay occurs in obtaining y_n. The bit rate will be equal to the

$$\min\left\{\frac{1}{t_m}, \frac{1}{t_\alpha}\right\} \qquad (4.3.29)$$

in the serial mechanization, and the same for the parallel mechanization because of the inherent pipeline structure as claimed by B.Liu et.al. in these cases.

In conclusion, B.Liu et.al. have found a means of implementing an Nth order difference equation without multipliers, but only add-and-shift operations. This has led to an important breakthrough in the hardware implementations of digital filters, and what remains now is to replace the conventional adders by some suitable algorithm. Three objectives are now borne in mind:

(1) To replace conventional adders as far as possible by a reasonable amount of memory to increase the speed of operation.
(2) To overcome the fundamental limitation in speed of conventional adders: the carry propagation path.
(3) To invest and capitalize on recent advances in semiconductor memory technology to implement the above two steps in LSI so as to reduce cost, power consumption and chip count in digital filters implementations.

In the next section, we shall look into existing addition processesperformed with memory technique and their achievement so far.

4.4 REPLACEMENT OF ADDERS BY MEMORY TECHNIQUE:

Existing methods in the use of memory in addition can be found in Sypherd's paper (70) where a serial-mode ROM adder has been proposed, as shown in figure 4.4.1. Here, the serial multiple input adder is readily and efficiently mechanized with a ROM adder. Data through the ROM adder is serial, thus n bits must be added at any one time. The manner in which the ROM performs the addition

process is largely dependent on the size of n. Consider the 256x4-bit ROM organization which requires 8-bit addresses. In this case, the eight address bits represent the numbers to be added in serial. They are five data bits and three carry bits. The memory output will be the serial sum plus carries. As eight bits are added together, therefore a maximum carry (100) will require three bits to represent it, and hence the three output carry bits.

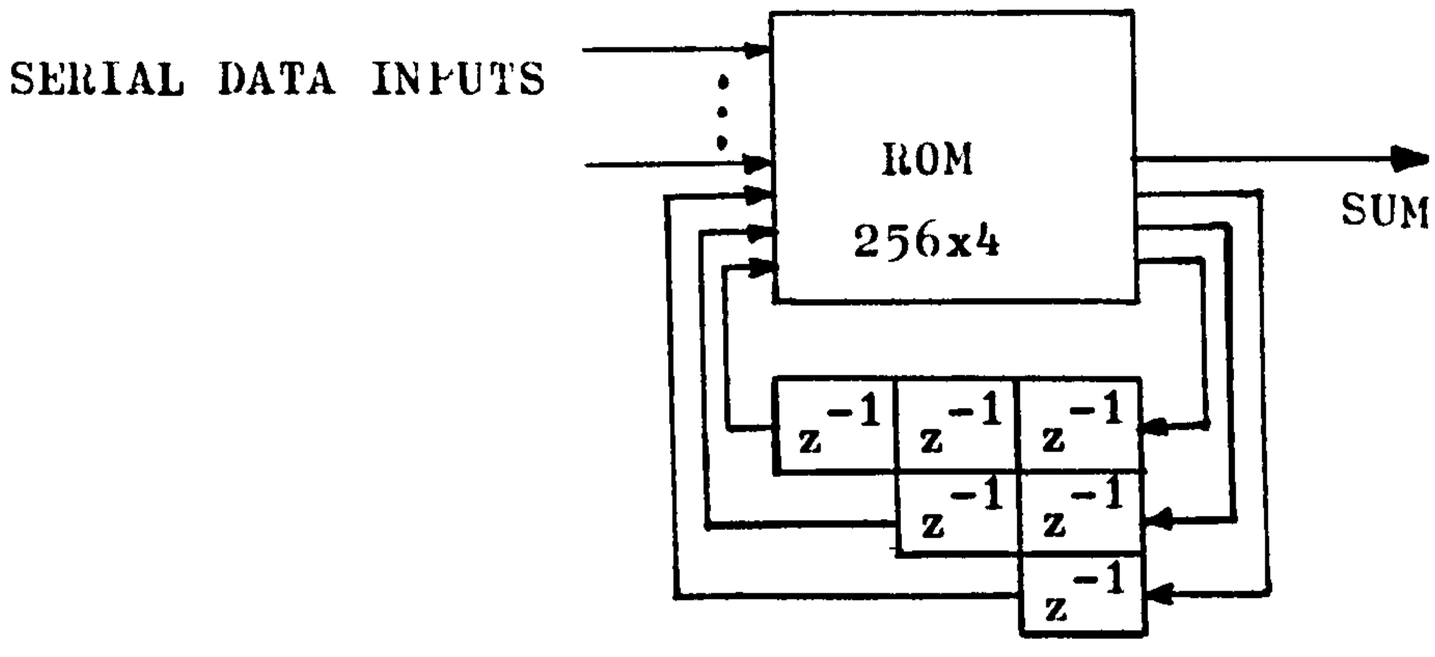


FIG: 4.4.1 SERIAL ROM ADDER MECHANISM
FOR FIVE OR LESS INPUTS

Using conventional adders, the summing of five inputs serially can also be implemented with seven full adders and the above six delay elements. Hence, here the tradeoff is seven full adders for 256x4 bits of memory. While the speed of the serial ROM adder depends very much on the wordlengths of the input data, the parallel ROM adder implementation proves impractical due to the excessive amount of memory required. Therefore, for high-speed applications, additions have still to be performed with high-speed adders, such as the full carry-lookahead adders, which comtemporary memory adders have not been able to replace.

4.5 CONCLUDING REMARKS:

Despite the recent introduction of the "combinatorial digital filters" which use memory to replace multiplier hardware, a lot of present designs still employ conventional multipliers due to the vast experience in exploiting them accumulated since the early days of digital signal processing.

Conceptually, filter hardware designs using conventional multipliers and memory techniuqe in multiplication differ in their implementations in the sense that the partial products in the case of the memory technique are precalculated and stored, whereas those in the case of conventional multipliers are not. Consequently, the algorithms employed in partial products production are different in both cases. In particular, B.Liu et.al. have proposed a "bit-sliced" technique to perform an Nth order difference equation, corresponding to an Nth order digital filter, without the need of conventional multipliers. In addition, their algorithm also makes use of the fact that the filter coefficients are constants, thereby reducing the total number of partial products significantly. As a result of comparing their "combinatorial filter" approach with other existing designs, B.Liu et. al. have claimed that their approach has made possible speeds of operation which cannot be achieved by existing hardware implementations of digital filters. Furthermore, their approach capitalizes on recent advances in semiconductor memory technology and is shown to be offering significant reductions in cost and power consumption for the same speed of operation as that of existing implementations, (74),(75).

Nevertheless, contemporary designs, both using multipliers and memory techniques, rely on conventional adders to sum their partial products and are therefore subject to the fundamental limitation in addition speed: the carry propagation path. However, in the next chapter, we shall describe in detail a novel hardware implementation of digital filters for high-speed realtime filtering which has employed an algorithm to overcome the above fundamental limitation in addition speed and to perform multiplications via an efficient use of memory. The three objectives (mentioned in section 4.3.2) in replacing conventional adders in the summation of partial products have been observed, and as a direct consequence, the proposed approach makes possible operation speeds which compare more favourably with B.Liu et.al. approach.

CHAPTER FIVE

A NOVEL HARDWARE IMPLEMENTATION OF DIGITAL FILTERS FOR HIGH-SPEED REALTIME FILTERING

5.1 INTRODUCTION:

In this chapter, we propose an algorithm for the hardware implementation of a general Nth order digital filter represented by the Nth order difference equation

$$y_n = \sum_{k=o}^{N} a_k . x_{n-k} - \sum_{k=1}^{N} b_k . y_{n-k} \qquad (5.1.1)$$

where the digital sequences $\{x_n\}$ and $\{y_n\}$ are respectively the input and output sequences of the digital filter, expressed to infinite precision. Similarly, the filter coefficients $\{a_k\}$ and $\{b_k\}$ are also expressed to infinite precision.

However, in practice, all digital processors are of finite precision (finite word length effects on the proposed hardware implementation of digital filters will be discussed in chapter six), and equation (5.1.1) can be rewritten as

$$y'_n = \sum_{k=o}^{N} a'_k . x'_{n-k} - \sum_{k=1}^{N} b'_k . y'_{n-k} \qquad (5.1.2)$$

where the input sequence $\{x'_n\}$ and the output sequence $\{y'_n\}$ are now expressed to finite precision along with the finite-precision filter coefficients $\{a'_k\}$ and $\{b'_k\}$.

As the speed of operation of a digital filter increases, either to permit realtime processing of wide-band signals or to time-share its arithmetic unit, there is a rapid increase in hardware complexity as measured by the number of integrated circuits used and in power consumption. In conventional implementations of digital filters, the major factor causing this increase lies with the high-speed adders and multipliers used. In other contemporary "multiplierless" approaches using memory techniques, the main factor remains with the high-speed adders used. In the above cases, the high-speed adders and multipliers rely on conventional designs which are limited in speed due to the carry propagation path in an addition process.

However, the proposed approach will not include any "adds" and "multiplies" as in conventional approaches, but instead employs a novel algorithm, hereafter known as the "Carry Brought Forward Compensation Scheme" (CBFCS), which brings together the low cost of semiconductor memory and the effectiveness of standard 4-bit full adders modified as "functional units" to implement an "Adderless-Multiplierless-Unit", to replace conventional computational units. Since these 4-bit full adders no longer behave as conventional adders, but wired as combinational logic units instead, the carry propagation path associated with an N-bit addition is effectively being "sectioned" down from a maximum of N bits long to a maximum of four bits only. Consequently, the proposed approach makes possible operation speeds which cannot be achieved by existing approaches in the hardware implementations of digital filters.

In short, the proposed approach aims to provide an algorithm in hardware implementation which enables an economical high-speed digital filter to be constructed for time-multiplexing of the arithmetic unit in realtime applications involving a large number of channels or wide-band signal processing. It capitalizes on recent advances in semiconductor memory technology to offer significant reductions in cost and power consumption for the same speed of operation as that of existing approaches. In replacing conventional multipliers and adders with memory as far as possible, the fundamental limitation in speed due the carry propagation path associated with an addition has been overcome to make possible operational speeds which cannot be obtained with existing approaches for a given technology, for example TTL, ECL etc.

5.2 THE CARRY BROUGHT FORWARD COMPENSATION SCHEME (CBFCS):

Equations(5.1.1) and (5.1.2) represent a general Nth order digital filter and if implemented directly will lead to instability for large N due to finite wordlength effects in digital filtering (44). However, as mentioned previously, an Nth order filter is in practice almost invariably implemented as a cascade or parallel combination of first- and second-order filters.

In this section, we present the "Carry Brought Forward Compensation Scheme" by concentrating on a general second-order

difference equation, while bearing in mind that the same approach holds for the direct implementation of an Nth order digital filter. Since a general Nth order digital filter is often implemented as a combination of first- and second-order filter sections, the results presented here constitute, in a sense, a complete picture of an Nth order implementation in the cascade or parallel form.

Consider a general second-order difference equation below:

$$y_n = a_o.x_n + a_1.x_{n-1} + a_2.x_{n-2} - b_1.y_{n-1} - b_2.y_{n-2} \qquad (5.2.1)$$

with $a_2 = 0$ in the case of a parallel realization.

Assume that all signals are bounded by ± 1. Let M bits including sign in two's complement notation be used to represent the filter variables $\epsilon \{x_n, x_{n-1}, x_{n-2}, y_{n-1}, y_{n-2}\}$ and N bits including sign for the filter coefficients $\epsilon \{a_o, a_1, a_2, b_1, b_2\}$ as follows:

$$x_k = -x_k^o + \sum_{j=1}^{M-1} x_k^j.2^{-j} \qquad x_k^j \epsilon \{0,1\} \qquad (5.2.2)$$

$$y_k = -y_k^o + \sum_{j=1}^{M-1} y_k^j.2^{-j} \qquad y_k^j \epsilon \{0,1\} \qquad (5.2.3)$$

where x_k^o and y_k^o are sign bits and are given a negative weighting due to the two's complement notation. Similarly,

$$a_k = -a_k^o + \sum_{i=1}^{N-1} a_k^i.2^{-i} \qquad a_k^i \epsilon \{0,1\} \qquad (5.2.4)$$

$$b_k = -b_k^o + \sum_{i=1}^{N-1} b_k^i.2^{-i} \qquad b_k^i \epsilon \{0,1\} \qquad (5.2.5)$$

where a_k^o and b_k^o are sign bits and are given a negative weighting due to the two's complement notation. Substituting equations (5.2.2) and (5.2.3) in (5.2.1), we have,

$$\begin{aligned} y_n = &-(x_n^o.a_o + x_{n-1}^o.a_1 + x_{n-2}^o.a_2 - y_{n-1}^o.b_1 - y_{n-2}^o.b_2) \\ &+ \sum_{j=1}^{M-1} 2^{-j}(x_n^j.a_o + x_{n-1}^j.a_1 + x_{n-2}^j.a_2 - y_{n-1}^j.b_1 - y_{n-2}^j.b_2) \end{aligned} \qquad (5.2.6)$$

Now equation (5.2.6) can be rewritten in a simpler form as

$$y_n = -\bar{y}_n^0 + \sum_{j=1}^{M-1} \bar{y}_n^j . 2^{-j} \qquad (5.2.7)$$

where y_n can be regarded as a final product formed by the summation of M properly shifted partial products, each of N bits, such that the jth partial product is of the form

$$\bar{y}_n^j = x_n^j . a_o + x_{n-1}^j . a_1 + x_{n-2}^j . a_2 - y_{n-1}^j . b_1 - y_{n-2}^j . b_2 \qquad (5.2.8)$$

which is positive for j=1 to j=M-1 and negative for j=0. Hence, the jth partial product in equation (5.2.8) is readily observed to be the sum of the jth bits of the filter variables multiplied by their respective filter coefficients. Each partial product is then right-shifted j places due to the weighting factor 2^{-j} in equation (5.2.7) above.

Let the summation of the M properly shifted partial products in equation (5.2.7) above be performed in a sequential manner such that two partial products are added at a time, starting with the least significant pair first. Consider next the summation of the least significant pair of properly shifted partial products which can be written as

$$2^{-(M-2)} \left[2^{-1} . \bar{y}_n^{M-1} + \bar{y}_n^{M-2} \right]$$

We next byte-slice the above two terms in the square bracket into two byte-sliced numbers, starting from the most significant end, so that each byte-sliced number is of the form

$$\gamma = \sum_{k=1}^{\lceil N/B \rceil} \gamma_k . 2^{-(k-1)B} \qquad (5.2.9)$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, and B is the number of bits in each byte.

Hence, the summation of the properly shifted (M-1)th and (M-2)th partial products has now become the summation of two byte-sliced numbers instead, such that bytes of the same significance are summed together in pairs. We further stipulate that a possible

carry-out of the summation of a pair of less significant bytes will not be passed on to be summed with the next pair of more significant bytes. Instead, these carry-outs from the summation of the (M-1)th and (M-2)th partial products, while properly weighted, will then be "brought forward" to be summed with the next partial product, i.e. the (M-3)th partial product, to form a "compensated partial product"; thus, leaving behind a "carry-depleted sum" of the previous (M-1)th and (M-2)th partial products. In the next summation cycle, this "compensated partial product", which will also be byte-sliced, will then be summed with the previously obtained "carry-depleted sum". This procedure will be repeated sequentially until a "depleted" form of y_n is reached, which will be further compensated to form y_n as explained below.

Assuming (for convenience) each of the above "compensated partial product" is an N-bit fraction, we define a function f with $\left[\frac{N}{B}-1\right]+5$ binary arguments as follows:

$$f(r_1^j, r_2^j, r_3^j, r_4^j, r_5^j, C_B^{j+1}, C_{2B}^{j+1}, \ldots\ldots, C_{\left[\frac{N}{B}-1\right]B}^{j+1})$$

$$= a_o.r_1^j + a_1.r_2^j + a_2.r_3^j - b_1.r_4^j - b_2.r_5^j + C_B^{j+1}.2^{-B} + C_{2B}^{j+1}.2^{-2B} + \ldots\ldots + C_{\left[\frac{N}{B}-1\right]B}^{j+1}.2^{-\left[\frac{N}{B}-1\right]B} \quad (5.2.10)$$

where the C_{kB}^{j+1} terms are carry-outs, such that the above mentioned "depleted" form of y_n can now be written as:

$$f_n = -f(x_n^o, x_{n-1}^o, x_{n-2}^o, y_{n-1}^o, y_{n-2}^o, C_{B,n}^1, C_{2B,n}^1, \ldots, C_{\left[\frac{N}{B}-1\right]B,n}^1) + \sum_{j=1}^{M-1} 2^{-j}.f(x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}^j, y_{n-2}^j, C_{B,n}^{j+1}, \ldots, C_{\left[\frac{N}{B}-1\right]B,n}^{j+1}) \quad (5.2.11)$$

Now, the function f above provides the necessary "compensated partial products" in the byte-sliced summation of f_n in the proposed "Carry Brought Forward Compensation Scheme". In computing f_n, a possible carry-out of the sum of the sum of the (k+1)th pair of B-bit bytes in the byte-sliced summation of the compensated (j+1)th partial product and the "carry-depleted sum" of the previous (M-j-2) less significant partial products is designated as $C_{kB,n}^{j+1} \in \{0, 1\}$.

As mentioned above, f_n, as given by equation (5.2.11), represents a "depleted" form of the final product y_n due to its "depleted" term

$$-f\ (x_n^o,\ x_{n-1}^o,\ x_{n-2}^o,\ y_{n-1}^o,\ y_{n-2}^o,\ C_{B,n}^1,\ \ldots\ldots,\ C_{\left[\frac{N}{B}-1\right]B,n}^1\ )$$

In order to further compensate the above term, we define the following two compensation functions.

$$f' = C_B^1.2^{-(B-1)} + C_{2B}^1.2^{-(2B-1)} + C_{3B}^1.2^{-(3B-1)} + \ldots\ldots + C_{\left[\frac{N}{B}-1\right]B}^1.2^{-\left(\left[\frac{N}{B}-1\right]B-1\right)} \qquad (5.2.12)$$

$$f'' = C_B^o.2^{-(B-1)} + C_{2B}^o.2^{-(2B-1)} + C_{3B}^o.2^{-(3B-1)} + \ldots\ldots + C_{\left[\frac{N}{B}-1\right]B}^o.2^{-\left(\left[\frac{N}{B}-1\right]B-1\right)} \qquad (5.2.13)$$

such that the final product is now given by

$$y_n = f_n + f_n' + f_n'' \qquad (5.2.14)$$

where f_n' and f_n'' are values of the compensation functions f' and f" respectively in the calculation of the nth output sample y_n, that is the final product as given in equation (5.2.14).

The functions f' and f" together constitute the compensation part of the "Carry Brought Forward Compensation Scheme" while the function f forms the carry brought forward part which provides the "carry-compensated partial products" of f_n, as opposed to the original partial products of y_n in equation (5.2.7). In the above sequential "byte-sliced" summation process of the proposed "Carry Brought Forward Compensation Scheme", carry-outs are continuously "brought forward" while "depleted sums" are left behind to be summed with "compensated partial products" in a "byte-sliced" manner until f_n is reached, which is then further compensated by the values f_n' and f_n'' to obtain the final product y_n.

Thus, the set of functions, {f, f', f"}, forms the "Carry Brought Forward Compensation Scheme" which is the proposed algorithm

in the hardware implementation of a general Nth order digital filter. The algorithm is now best illustrated by way of examples.

Consider equation (5.2.1) with all filter coefficients made zero, except a_o, such that $y_n = a_o.x_n$. In general, the proposed algorithm is independent of the four different cases of the signs of a_o and x_n, viz., (+,+), (+,-), (-,+), (-,-). In our example, we shall assume the worst case when both a_o and x_n are negative and have the value -0.125 so that their decimal product y_n = 0.015625.

Prior to applying the proposed "Carry Brought Forward Compensation Scheme" in the above example, we now work out the product y_n by straightforward two's complement arithmetic as below:

		1.111	MULTIPLICAND = -0.125
		x 1.111	MULTIPLIER = -0.125
		1.1111	RIGHT SHIFT
		1.111	ADD MULTIPLICAND
		1.11101	RIGHT SHIFT
		1.111	ADD MULTIPLICAND
		1.111001	RIGHT SHIFT
		0.001	SUBTRACT MULTIPLICAND
IGNORE CARRY	(1)	0.000001	PRODUCT = 0.015625

The above example has illustrated the calculation of the final product y_n by applying equation (5.2.7). We shall now show how equations (5.2.10), (5.2.12) and (5.2.13) can be made use of in applying the proposed algorithm in obtaining y_n for the above example. Since N=4 in the above example, we shall choose B=2, i.e. the partial products are byte-sliced into numbers consisting of 2-bit bytes.

		1.111	FILTER COEFFICIENT
		x 1.111	FILTER VARIABLE
		1.1 11 1	RIGHT-SHIFTED PARTIAL PRODUCT
CARRY-OUT		1.111 1	ORIGINAL PARTIAL PRODUCT
		1.1 01 01	DEPLETED SUM (RIGHT-SHIFTED)
		0.0 01	COMPENSATED PARTIAL PRODUCT
	B'→	1.1 11 00 1	DEPLETED SUM (RIGHT-SHIFTED)
CARRY-OUT	B →	0.0 01	SUBTRACT ORIGINAL PARTIAL PRODUCT
		1.1 00 00 1 ←	f_n
		0.1 ←	f''_n
IGNORE CARRY	(1)	0.0 00 00 1 ←	y_n

Here f''_n is defined as the value of the compensation function f" in the calculation of the nth output sample y_n by equation(5.2.14).

The operation of the proposed algorithm may be explained via the following sequential byte-sliced summation procedure:

STEP 1: This step involves the byte-sliced summation of the 2nd and 3rd partial products of f_n as in equation (5.2.11) with M=4. Before it is byte-sliced, the 3rd partial product is first right-shifted by one bit with respect to the 2nd. Since both $C^4_{2,n}$ and $C^3_{2,n}$ are zero in this step, no compensation is required, and the 2nd and 3rd partial products of f_n are the same as the original partial products of y_n.

STEP 2: In step one above, a carry-out $C^2_{2,n}$ is generated which is then properly weighted by the factor 2^{-2}, and brought forward to be summed with the 1st partial product of y_n to obtain a compensated partial product of f_n as follows:

```
  1.111    ORIGINAL 1ST PARTIAL PRODUCT
+ 0.010    CARRY-OUT BROUGHT FORWARD
  0.001    COMPENSATED PARTIAL PRODUCT
```

In this step, the above compensated partial product is then summed with the depleted sum formed in step one. Again, this depleted sum has to be right-shifted to restore the relative weighting with respect to the compensated 1st partial product of f_n. A second depleted sum also results in this step.

STEP 3: As the value of $C^1_{2,n}$ is zero, the 0th partial product of f_n is the same as that of y_n. However, as in equation (5.2.11), the 0th partial product carries a negative weight due to the two's complement notation. Consequently, the 0th partial product is now subtracted from the depleted sum of the previous step instead.

STEP 4: A carry-out $C^0_{2,n}$ is generated in step three so that the resultant f_n has to be further compensated by f''_n as shown. The compensated value of f_n is thus the final product y_n.

In the above example, the compensation function f' is not required. In order to show the function of f', we shall now reconsider this example and assume that the "Carry Brought Forward Compensation Scheme" has been performed in the first two steps.

```
                    1.111
                  x 1.111
                  1.1111
                   1.111
                1.1 11 01
                1.1 11 11
CARRY-OUT
A' ------>      1.1 01 00 1     DEPLETED SUM (RIGHT-SHIFTED)
A  ------>      1.1 11 11       SUBTRACT COMPENSATED PARTIAL PRODUCT
                  0.1           <-- f'n
CARRY-OUT
          (1)   1.1 00 00 1     <-- fn + f'n
                  0.1           <-- f"n
          (1)   0.0 00 00 1     <-- yn = fn + f'n + f"n
```

In the above steps, end-carries in brackets are ignored as in the previous example. In step three, the value of $C^1_{2,n}$ is 1, that is a carry-out, therefore, the 0th partial product of f_n is a "compensated partial product", and the two's complement of which is added here due to the negative weighting of the 0th partial product. Since $C^1_{2,n}=1$, a further compensation of f_n by a value f'_n is needed at this stage. In the fourth step, the value of f_n is again compensated, this time by a value f''_n, due to the carry-out $C^o_{2,n}=1$.

From the above example, it is obvious that the introduction of the compensation function f' is really due to the subtraction required in step three. To justify this, we again examine the last two examples and compare results in the third step where the function f' is introduced. From the working of the above examples, we have

$$\begin{aligned} A' + A + f'_n &= B' + B \\ &= 1.111001 + 0.001 \\ &= A' + 0.010 + 0.001 \end{aligned}$$

such that

$$\begin{aligned} f'_n &= 0.010 + 0.001 - A \\ &= 0.010 + 0.001 + 0.001 \\ &= 0.100 \end{aligned}$$

which is the required value as given by equation (5.2.12).

We further note that the value of f'_n is always equal to 0.100 in a 4x4 multiplication, that is $f'_n = C^1_{2,n}.2^{-1}$, irrespective of the values of the multiplicand and multiplier, as B'-A' = B-A = 0.010 and f'_n = (B'-A') + (B-A) = 0.100 where A is related to B in the same way as A' is related to B'.

To conclude this section, we shall now rewrite equation (5.2.11) in a manner similar to equation (5.2.7) as follows:

$$f_n = -\tilde{f}_n^0 + \sum_{j=1}^{M-1} \tilde{f}_n^j . 2^{-j} \qquad (5.2.15)$$

where f_n can be regarded as a "depleted" form of y_n, formed by the summation of M properly shifted partial products, each of N bits and compensated by the carry-outs brought forward, such that the jth "compensated partial product" is of the form

$$\tilde{f}_n^j = x_n^j . a_o + x_{n-1}^j . a_1 + x_{n-2}^j . a_2 - y_{n-1}^j . b_1 - y_{n-2}^j . b_2 + C_{B,n}^{j+1} . 2^{-B} + C_{2B,n}^{j+1} . 2^{-2B} + \ldots\ldots + C_{\left[\frac{N}{B}-1\right]B,n}^{j+1} . 2^{-\left[\frac{N}{B}-1\right]B} \qquad (5.2.16)$$

which is positive for j=1 to j=M-1 and negative for j=0.

The above summation of the "compensated partial products" of f_n is implicitly assumed to be in a byte-sliced manner previously described. By the introduction of the function f, the summation of the original partial products of the final product y_n in equation (5.2.7) has been modified to become that as given by f_n in equation (5.2.15) above, where the value of f_n is further compensated by f' and f", the compensation functions. In essence, the proposed algorithm, the "Carry Brought Forward Compensation Scheme", has employed a "bit-sliced" approach in calculating the essential partial products of a general difference equation, which are subsequently summed in a "byte-sliced" manner. In this manner, the proposed algorithm has the advantage of simultaneously performing the 2N+1 multiplications required in a general Nth order difference equation, while overcoming the fundamental problem of carry propagation in the addition of these multiplication products. In the byte-sliced summation of the above N-bit "compensated partial products", carry propagation has been confined to within bytes of B-bit long, thereby effectively "sectioning" the worst case N-bit carry propagation path down to a maximum of B bits long. In the next section, we shall consider the implementation of a general

second-order difference equation by the "Carry Brought Forward Compensation Scheme" using memory hardware and standard 4-bit full adders modified as logic function units as required in the above scheme.

5.3 IMPLEMENTATION OF A GENERAL SECOND-ORDER DIFFERENCE EQUATION BY THE CARRY BROUGHT FORWARD COMPENSATION SCHEME (CBFCS):

5.3.1 INTRODUCTION:

The function f in equation (5.2.10) can only take on a maximum of

$$2^{\left(\left\lceil\frac{N}{B}-1\right\rceil+\ 5\right)} \qquad (5.3.1)$$

distinct values or binary states, depending on the binary vectors which form its arguments. The total number of arguments of f is given by

$$\left(\left\lceil\frac{N}{B}-1\right\rceil+\ 5\right) \qquad (5.3.2)$$

where, in the above equations, $\lceil x \rceil$ has the meaning of the smallest integer greater than or equal to x.

All the above possible binary states of f can be realized by a memory with its locations addressed by the arguments of f. Thus, equation (5.2.14) suggests a possible mechanization of equation (5.2.1). If we further let the value of B be equal to, or a multiple of four, then we can employ memory and standard 4-bit full adders hard-wired as logic function units to implement the "Carry Brought Forward Compensation Scheme".

The content of the CBFCS memory consists of precalculated values of all possible partial products of f_n which include the original partial products of y_n when the carry-outs $\epsilon \{C^j_{B,n}, \ldots\ldots\}$ are all zero, and the compensated values of the original partial products when not all of the above carry-outs are equal to zero. The partial products of f_n have been referred to as the "compensated partial products" as opposed to the uncompensated original partial products of y_n.

As previously mentioned, the standard 4-bit full adders will not be used as conventional adders. In fact, they are used as logic function units to generate the carry-outs $\in\{C^j_{B,n}, C^j_{2B,n},\ldots.\}$ of the function f, with their carry-in connections hard-wired to some logic circuits which implement the compensation function f'.

Figure 5.3.1 shows that the carry-outs $\in\{C^j_{B,n}, C^j_{2B,n},\ldots.\}$ are diverted to address the CBFCS memory to yield the partial products of f_n in the production of y_n. On the other hand, if these 4-bit full adders were used as conventional adders, then ripple carry propagation is required in between stages, as shown by the dotted arrows in figure 5.3.1, for B=4.

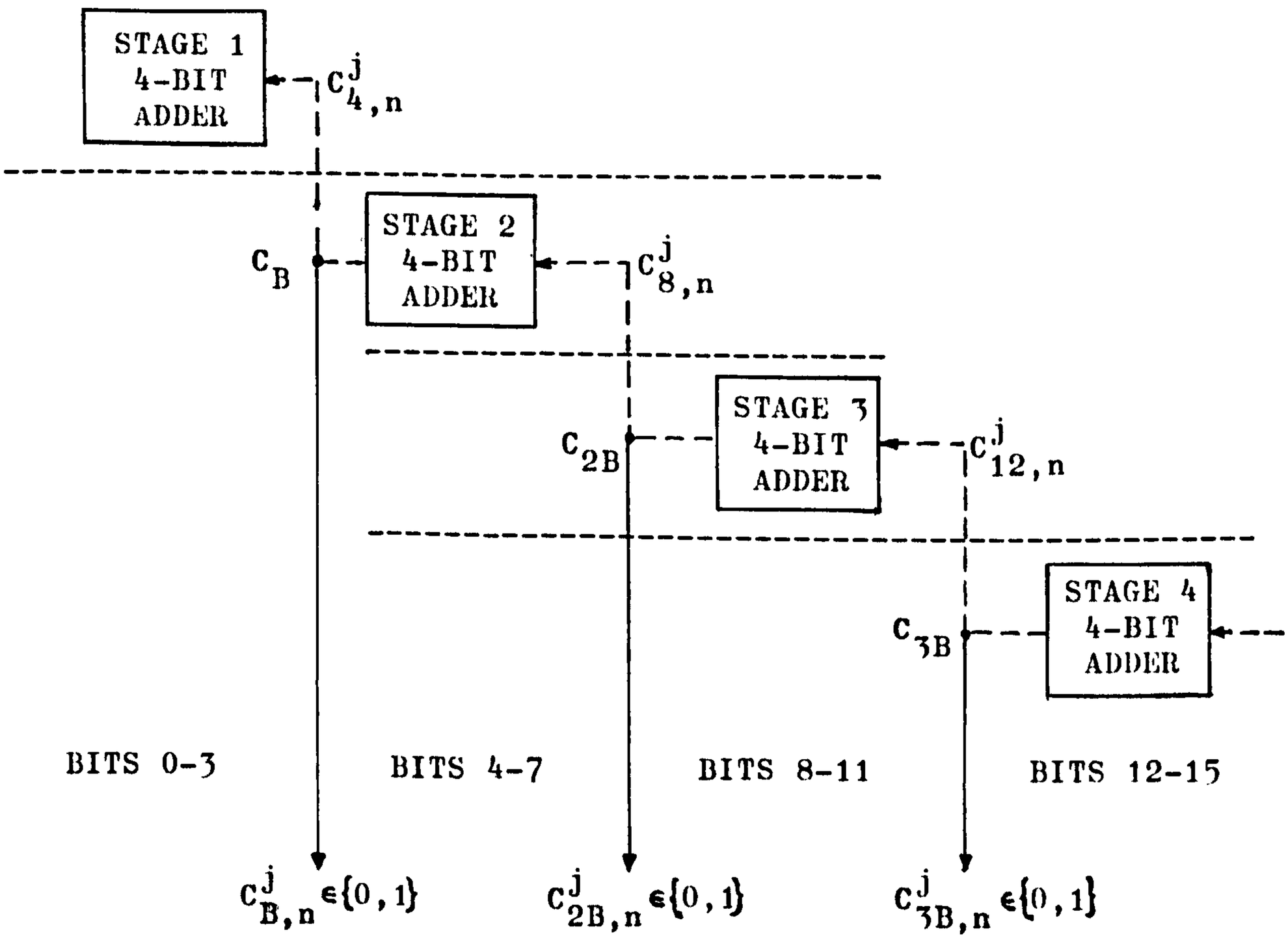


FIG: 5.3.1 SECTIONING OF THE CARRY PROPAGATION PATH SHOWN FOR B=4 IN THE CARRY BROUGHT FORWARD COMPENSATION SCHEME

In the proposed approach, the carry propagation path is essentially "sectioned" at the sectioning points $\in\{C_B, C_{2B}, \ldots.\}$ where the carry-outs $\in\{C^j_{B,n}, C^j_{2B,n},..\}$ are diverted to address the CBFCS memory. Therefore, the effective carry propagation time is

now t, where t is the carry propagation time within a B-bit byte. On the other hand, if the standard 4-bit full adders were used as conventional adders, the "unsectioned" carry propagation time for ripple carry between stages would have been $\lceil N/B \rceil t$ instead. Thus, the worst case N-bit carry propagation path is now effectively "sectioned" down to a maximum of only B bits long. Furthermore, the summation time of two consecutive "compensated partial products" in the proposed "byte-sliced" sequential summation is therefore equal to the time required to sum two B-bit bytes.

In figure 5.3.1 above, the "sectioning points" have been chosen to be $\{C_4, C_8, C_{12}, \ldots.\}$, such that t is actually the typical carry propagation time of a 4-bit full adder and the carry-outs of these adders are $\{C^j_{4,n}, C^j_{8,n}, \ldots\ldots\}$. In addition, if the typical addition time of a 4-bit full adder is s, then the summation time of two consecutive "compensated partial products" will be s instead of $s+\lceil (N/4)-1 \rceil t$ for $s<2t$ and $s+\lceil (N/4)-2 \rceil t$ for $s>2t$, where $\lceil N/4 \rceil$ is the number of 4-bit full adders used.

The above arrangement of standard 4-bit full adders as logic function units has been referred to, by us, as the "Adderless-Multiplierless-Unit" (AMU), while no conventional multiplication or addition is performed by the unit as such. The above "Adderless-Multiplierless-Unit" is a direct consequence of the proposed "Carry Brought Forward Compensation Scheme" which is used in conjunction with the CBFCS memory to implement the function f_n. Having briefly mentioned the AMU above, we shall now return to consider the memory requirement of the CBFCS memory in more details.

Consider the implementation of the general second-order difference equation given by equation (5.2.1). Assume that N=16 bits are used to represent the filter coefficients. "Sectioning" the carry propagation path at the "sectioning point $C_{B=8}$", would mean breaking the path into two B=8 bit bytes and the memory size required is shown in figure 5.3.2. For the sake of convenience, the CBFCS memory has been shown as two different memory blocks: the original memory required if $C^j_{8,n}=0$ and the extra memory needed if $C^j_{8,n}=1$ for the compensated partial products. We shall now calculate the actual CBFCS memory size required for the above case.

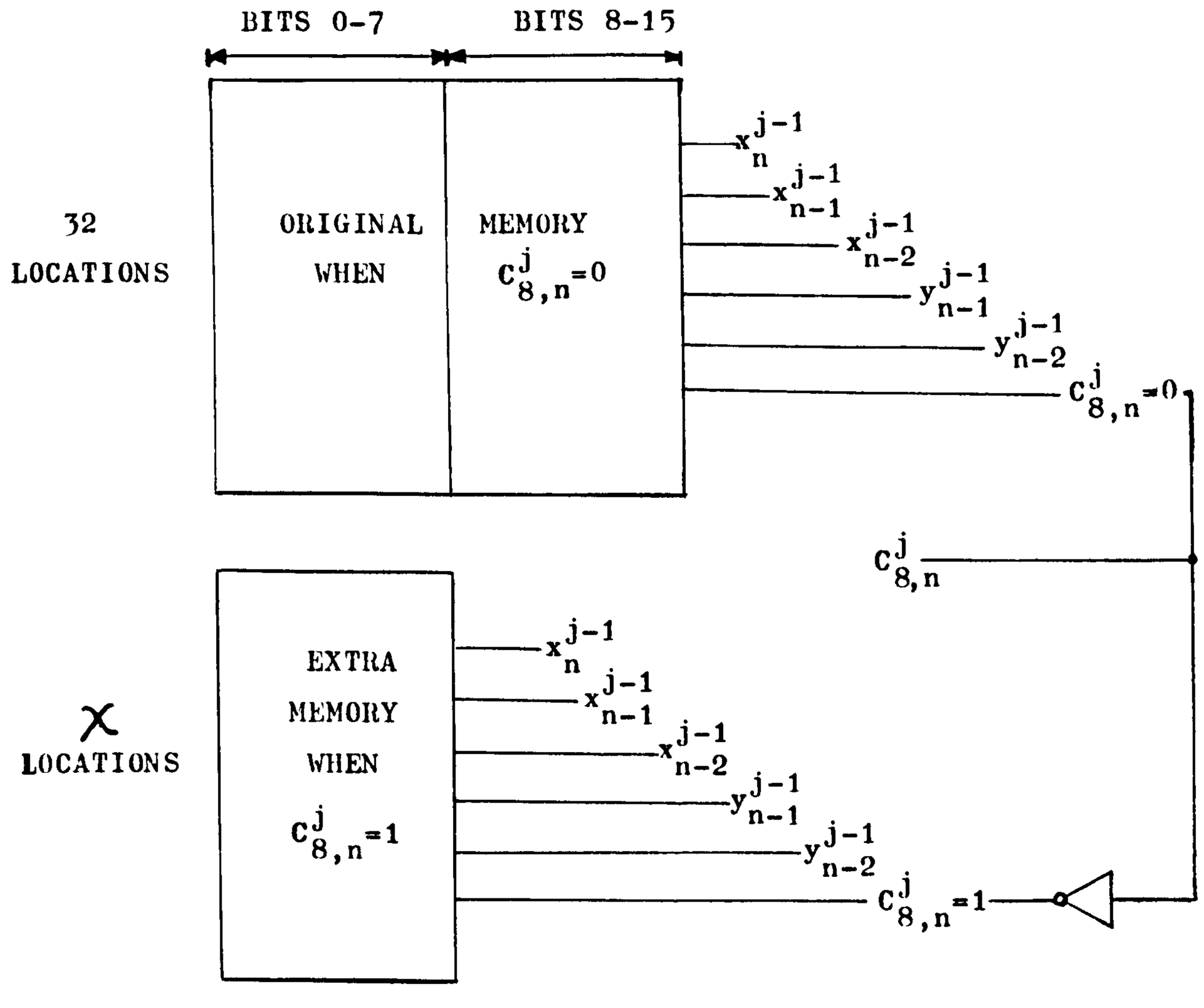


FIG: 5.3.2 CBFCS MEMORY REQUIREMENT FOR "SECTIONING" AT POINT $C_{B=8}$.

Equation (5.3.1) gives the maximum distinct locations or binary states of f. However, in practice, not all of these binary states have to be implemented in actual memory hardware. From figure 5.3.2, the original memory size required when $C^j_{8,n}=0$ is seen to be equal to $2^5 x16=32x16$ bits. This memory block will be addressed by the respective filter variable bits and $C^j_{8,n}=0$ as shown in the above figure. However, when $C^j_{8,n}=1$, the extra memory required will only be given by $\chi x8$, where χ is an integer less than or equal to 32. We shall elaborate on this in the next paragraph.

If $C^j_{8,n}=0$, then the output of the CBFCS memory will be a location within the original memory block. However, when $C^j_{8,n}=1$

the output of the CBFCS memory will be the last eight bits (8-15) of a location in the original memory, with the 8th bit inverted, and a corresponding location in the extra memory, which then comprises the first eight bits (0-7) of the output word. Moreover, from equations (5.2.8) and (5.2.16), the value of a "compensated partial product" of f_n is seen to be the sum of a corresponding original partial product of y_n and the carry brought forward $C^j_{8,n}=1$ weighted by the factor 2^{-8}. Hence, not all 32 locations in the original memory block will have their first eight bits changed after the addition of the $C^j_{8,n}=1$ carry. Consequently, the size of the extra memory block will merely amount to χ locations where $\chi_{max}=32$. This will be clarified by way of an example.

```
  1.1000100 01111111   ORIGINAL PARTIAL PRODUCT
+           1          WEIGHTED CARRY BROUGHT FORWARD
-------------------
  1.1000100 11111111   COMPENSATED PARTIAL PRODUCT
```

The above example shows that the first eight bits of the original partial product from the original memory **block** has remained unchanged after the addition of $C^j_{8,n}=1$ to form the "compensated partial product, the first eight bits of which do not have to be stored in this instance. Hence, from the above example, it is clear that unless the 8th bits of the 32 locations in the original memory are logical 1, the value of χ will, in general, be less than $\chi_{max}= 32$.

Since all partial products of f_n are precalculated, the content of the CBFCS memory will be known and therefore, the value of χ. Nevertheless, for the above case, the maximum increase in memory size, i.e. the extra memory required, is $\frac{1}{2}$ times that of the original memory, while the speed of operation has been doubled as compared to that of conventional full adders. Consequently, the tradeoff is made between the speed of addition and a reasonable increase in memory. Furthermore, for a general second-order difference equation, the value of χ can be estimated from the probability relation below:

χ = (Probability of the 8th bit of an arbitrary location in the original memory to be logical 1) x 32

In practice, the need of "sectioning" the carry propagation path into more than three B-bit bytes does not often arise. The table below shows the possible combinations of the set of carry-outs $\{C^{j+1}_{B,n}, C^{j+1}_{2B,n}, C^{j+1}_{3B,n}\}$ and the extra memory requirement.

$C^{j}_{B,n}$	$C^{j}_{2B,n}$	$C^{j}_{3B,n}$	EXTRA MEMORY
0	0	0	
0	0	1	χ x B
0	1	0	χ x 2B
0	1	1	χ x 2B
1	0	0	χ x 3B
1	0	1	χ x 3B
1	1	0	χ x 3B
1	1	1	χ x 3B
TOTAL 8 STATES			$4\frac{1}{4}\chi$N

Thus, when the carry propagation path is "sectioned" into four B-bit bytes, the maximum extra memory required is $4\frac{1}{4}$ times that of the original memory required. However, the increase in operational speed depends on the value of B and N. Therefore, in general, one has to compromise and choose the optimum "sectioning" of the carry propagation path to achieve a maximum increase in speed without an excessive increase in extra memory required.

5.3.2 A SERIAL IMPLEMENTATION OF THE CBFCS:

In the previous section, we have shown that equation (5.2.14) suggests a possible mechanization of a general second-order difference represented by equation (5.2.1). Figure 5.3.3 depicts a serial implementation of the second-order equation by the "Carry Brought Forward Compensation Scheme". In this general serial implementation, the carry propagation path in the "byte-sliced" summation of two N-bit "compensated partial products" of f_n is assumed to be "sectioned" every B bits, where B=4, to give the "brought forward" carry-outs $\epsilon\{C^{j}_{B,n}, C^{j}_{2B,n}, \ldots\ldots\}$, therefore

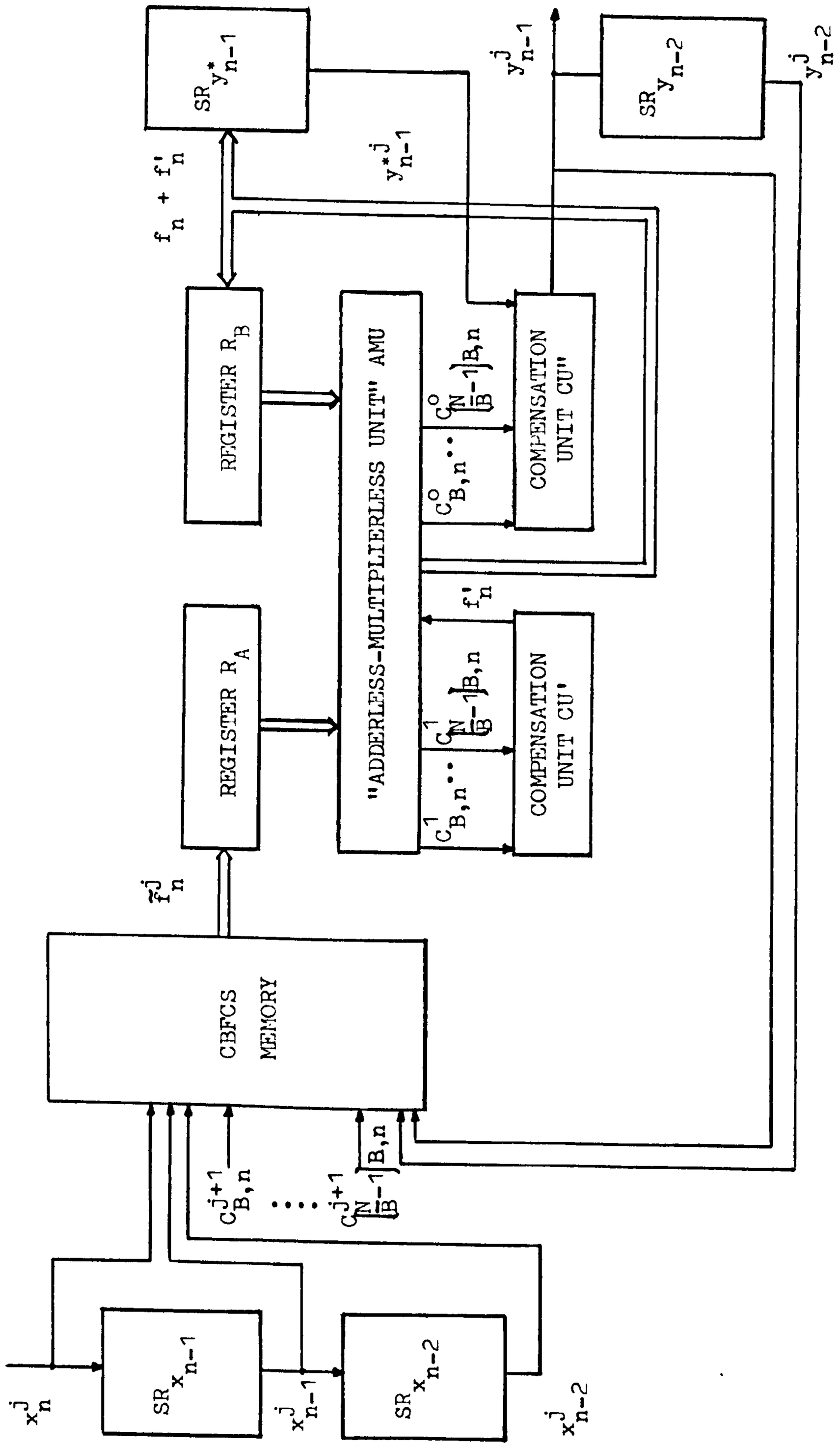


FIG: 5.3.3 A SERIAL IMPLEMENTATION OF THE "CARRY BROUGHT FORWARD COMPENSATION SCHEME".

requiring $\lceil N/B \rceil$ standard 4-bit full adders, hard-wired as logic function units in the "Adderless-Multiplierless-Unit" (AMU). The "compensation functions", f' and f", are generated by the logic units CU' and CU" respectively.

The operation of the serial implementation is illustrated in figure 5.3.3. M-bit data shift serially out of shift registers:

$$(SR_{x_{n-1}}, SR_{x_{n-2}}, SR_{y^*_{n-1}}, SR_{y_{n-2}})$$

with the least significant bits (M-1) leading. At each shift, a new vector:

$$(x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}^j, y_{n-2}^j, C_{B,n}^{j+1}, \ldots\ldots\ldots, C_{\left(\frac{N}{B}-1\right)B,n}^{j+1})$$

appears at the address input of the CBFCS memory which realizes the function f as given in equation (5.2.10). The output of the CBFCS memory is $\tilde{f}_n^j$, the jth "compensated partial product" of f_n, which is then loaded in parallel into register R_A. The output of register R_A is connected to one of the two summation inputs of the AMU. The other summation input of the AMU is hard-wired to the output of register R_B in parallel with one bit right shift to weight the "depleted sum" of the AMU summation output. As mentioned above, the AMU is made up of $\lceil N/B \rceil$ 4-bit full adders connected as logic function units. The carry outputs of these adders issue the set of carry-outs:

$$\{C_{B,n}^{j+1}, C_{2B,n}^{j+1}, C_{3B,n}^{j+1}, \ldots\ldots\ldots, C_{\left(\frac{N}{B}-1\right)B,n}^{j+1}\}$$

in parallel, which are brought forward to address the CBFCS memory, while the carry inputs of the above adders are connected to the output of the compensation unit CU'. The above set of carry-outs are also connected to the inputs of the compensation units CU' and CU", such that their values are clocked into the appropriate units at the desired moment as described below.

The operation of this serial implementation repeats every M+1 clock cycles. In the first M clock cycles, the "compensated partial products" of f_n are summed in the proposed "byte-sliced"

manner while a "byte-sliced" subtraction is performed in the (M+1)th or last clock cycle. After M such "byte-sliced" summations, the value f'_n of the compensation function f' is produced and clocked into the compensation unit CU'. The value of f_n is then first compensated by the value f'_n in the (M+1)th clock cycle when the above mentioned "byte-sliced" subtraction is performed, yielding the value $f_n + f'_n$ at the output of the AMU. The value f''_n of the compensation function f" is also produced at the end of the (M+1)th clock cycle. While the value $f_n + f'_n$ is clocked into the shift register $SR_{y^*_{n-1}}$, the value f''_n is simultaneously clocked into the compensation unit CU" at the start of the next (M+1) clock cycles of the filter operation. At each data shift in the next (M+1) clock cycles, the content of the shift register $SR_{y^*_{n-1}}$ is then shifted serially into the compensation unit CU" and y^*_{n-1} is bit by bit compensated by the compensation function f" to yield the output

$$y^j_{n-1} = (f_{n-1} + f'_{n-1} + f''_{n-1})^j$$

which is the serial delayed version of the previous output sample.

The compensation process by the function f" is realized in hardware with one logic gate level and consequently, the process can be performed in some "dead time" of the system, while not directly interfering with other operations carried out concurrently. Rounding of the output sample is done at the start of every (M+1) clock cycles, that is pre-rounding is assumed. This has the advantage of saving at least one clock cycle in the actual hardware implementation. The introduction of the register R_A enables concurrent operations of memory access of the CBFCS memory and the "byte-sliced" summation of the compensated partial products of f_n. However, it should be noted that due to the register R_A, (M+1) clock cycles are now required to yield an output sample y_n which otherwise could have been produced in M clock cycles. Nevertheless, the above concurrent operations greatly improves the operational speed of the resultant filter at the expense of an extra clock cycle. As a result, when realizing a recursive difference equation (corresponding to a feedback structure), the above serial implementation is hereafter referred to as a "pseudo-pipeline" structure while its operation

will be genuinely pipelined when realizing a non-recursive difference equation (corresponding to no feedback paths). Thus, in contrast to the recursive case, the serial implementation only requires M clock cycles in producing an output sample y_n for a non-recursive filter.

Having described the operation of the serial implementation, some practical comments are now in order. In figure 5.3.3, the necessary control circuitry is assumed and consequently not shown. In addition, any possible end-overflows in the first M "byte-sliced" summation have to be corrected by an "End-Overflow Correction Unit" (not shown in figure 5.3.3) which will be described in chapter seven. Furthermore, as has already been pointed out in section 4.3.2, when right-shifting two's complement numbers, modifications have to be made in order to restore the proper weighting of the number being shifted. This is also performed by the above "End-Overflow Correction Unit" which thus serves a dual-purpose. Finally, if the coefficients of the difference equation have to be scaled in order to be accommodated in the CBFCS memory, then appropriate re-scaling of the filter output sample has to be performed before the filter can be used to compute the next output sample. This is performed by a "Scaling Correction Unit" described in chapter seven.

We shall now examine the speed of operation of this serial implementation of the proposed CBFCS algorithm. Essentially two operations have to be completed before a new data bit can be shifted into the system. These operations are:

(1) Evaluation of the function f. The time taken is the CBFCS memory access time t_m.

(2) "Byte-sliced" summation of two N-bit "byte-sliced" numbers. The time required is the addition time of the AMU, t_a.

By the introduction of R_A, the above serial implementation becomes a "pseudo-pipeline" structure as previously mentioned, and the two operations can be performed concurrently. Consequently, the throught-put rate of the serial processor is:

$$\frac{1}{t_s} = \min\left\{\frac{1}{t_m}, \frac{1}{t_a}\right\} \qquad (5.3.3)$$

with a clock period of

$$t_s = \max\left\{t_m, t_a\right\} \qquad (5.3.4)$$

where the symbol in equation (5.3.3) means the minimum of the two factors and the symbol in equation (5.3.4) stands for the maximum of the two factors.

Using Schottky ROM's/RAM's/PROM's (e.g. 74SNS200A, 74SNS189, 74SNS288 etc.), the CBFCS memory access time, t_m = 25nsec. For the example shown in figure 5.3.2, where N = 16, B = 8, that is "sectioning" the carry propagation path into two 8-bit bytes, the "byte-sliced" summation of two "byte-sliced" 16-bit numbers can be performed by the AMU in 15nsec and 23nsec using Schottky and ordinary TTL 4-bit full adders respectively. Hence, the maximum operational throught-put rate of the serial processor is 1/25 or 40 MHz when using Schottky TTL memory and standard TTL 4-bit full adders. This further implies that "ideally", the serial processor can be used to process an input signal at 20MHz bit rate.

The word rate of the serial implementation depends on whether the system is employed to implement a recursive difference equation as a "pseudo-pipeline" structure, or a non-recursive difference equation as a genuine pipeline structure. Therefore the word rate,

$$\text{for recursive difference equation} = \frac{1}{(M+1)t_s} \qquad (5.3.5)$$

$$\text{for non-recursive difference equation} = \frac{1}{Mt_s} \qquad (5.3.6)$$

The above difference in one extra clock cycle between the serial implementations of the recursive case and the non-recursive case is due to the inherent feedback in the former case, while the latter case has no feedback in its structure.

To conclude this section, we point out that the serial implementation of the CBFCS algorithm represents only the slow-speed realization of the proposed approach to the implementation of realtime digital filters. The high-speed version of the proposed approach will call for a parallel implementation of the CBFCS algorithm, which will be introduced in the next section. Depending on particular specifications and requirements of individual cases, a comprise between the slow-speed version and high-speed version can often be obtained, resulting in a medium-speed version.

5.3.3 A PARALLEL IMPLEMENTATION OF THE CBFCS:

In the high-speed parallel implementation, we shall apply the CBFCS algorithm proposed in the previous section to a parallel or simultaneous summation of all partial products of the final product y_n given by equation (5.2.7) as opposed to the sequential summation in the serial implementation. However, before we can apply the CBFCS algorithm, some modifications are required to restore the relative weightings of the partial products as shown in the example below:

	1.001	FILTER VARIABLE = -0.875
	x 1.111	FILTER COEFFICIENT = -0.125
	1.1\1001	ADD RIGHT-SHIFTED PARTIAL PRODUCT
	1.\1001	ADD RIGHT-SHIFTED PARTIAL PRODUCT
	1\1001	ADD RIGHT-SHIFTED PARTIAL PRODUCT
	\0.111	SUBTRACT PARTIAL PRODUCT
IGNORE CARRY	0.000111	FINAL PRODUCT y_n = 0.109375

It is obvious in the above example that the partial products cannot be summed directly as in pure (unsigned) binary or sign-and-magnitude representations. The 1's to the left of the dotted line are required to compensate the relative weightings of the partial products due to the two's complement representation after right shifting. As a result, more bits have to be summed which further implies an increased complexity of the addition process in terms of hardware and longer carry propagation paths. We shall therefore, examine the above situation in details and introduce a Sign Transformation to modify the CBFCS algorithm. Consider next, the above example with the numbers regarded as if they were pure binary numbers.

	(1).001	FILTER VARIABLE
	x (1).111	FILTER COEFFICIENT
	(1) 001	RIGHT SHIFT
	(1) 0 01	RIGHT SHIFT
	(1) 0 0 1	RIGHT SHIFT
	1(0)(0)(1)	RIGHT SHIFT
	0.0 0 0 111	PRODUCT y_n

Since the sign bits of two's complement numbers are given a negative weighting, therefore, the bits in brackets in the above example carry a negative weighting as well. It is further noticed that the previous extra 1's are no longer necessary if the positve

bits and the negative bits are summed seperately. The procedure of summing the positive and negative bits seperately can be illustrated by first re-grouping the bits as below:

$$
\begin{array}{r}
(1).001 \\
\underline{\times\ (1).111} \\
001 \\
0\ 01 \\
1\ 0\ \ 0\ \ 0\ 1 \\
0(1)(1)(1) \\
\underline{0(0)(0)(1)} \\
?\ ?\ \ ?\ \ ?\ 111
\end{array}
$$

Thus, the positive and negative bits have been seperated into two groups. Furthermore, the bracketed bits in the bottom two rows are in fact two negative binary numbers whose sum can therefore be subtracted from the sum of the positive bits in two's complement notation as shown below:

$$
\begin{array}{r}
(1).001 \\
\underline{\times\ (1).111} \\
001 \\
001 \\
1\ 0001 \\
(1)001 \\
\underline{(1)111} \\
0.000111
\end{array}
$$

IGNORE END-CARRY

In the above example, we further notice that there are two and only two negative numbers in two's complement notation, which are in fact, the filter coefficient and variable themselves. The positively weighted 1 in the end column cancels out with one of the two negatively weighted 1's, leaving only a negatively weighted 1 behind. In short, we have first treated the partial products of y_n as if they were pure (unsigned) binary numbers by noting the positive and negative weightings of individual bits as shown in the second example. Secondly, we have re-grouped the bits into positive and negative numbers as shown in example three. Finally, we have represented the negative numbers in two's complement notation such that the negatively weighted bits are now grouped together in the end column as shown in the last example.

The above procedure has enabled the partial products of y_n to be summed directly irrespective of the signs of the filter variable and coefficient. The extra 1's required in the first example are no longer necessary. Furthermore, the above procedure can be represented compactly by the Sign Transformation below:

STEP 1: Treat the partial products of y_n as it they were pure binary numbers and then reshuffle their bit structure as shown below:

1	0	0	1
1	0	0	1
1	0	0	1
1	0	0	1

STEP 2: Two's complement the two words in rectangles. Note that the bit inside the square is common to both words, and after two's complementations the words in the rectangles are now interchanged as shown below:

1	0	0	1
0	0	0	1
0	0	0	1
1	1	1	1

STEP 3: Apply a "shearing" to the above data block so as to restore the previous relative weightings of the partial products and perform a straightforward binary summation. The sum is then the final product y_n in two's complement notation.

			1	0	0	1	
		0	0	0	1		
	0	0	0	1			
+	1	1	1	1			
(1)	0	. 0	0	0	1	1	1

IGNORE END-CARRY — FINAL PRODUCT y_n

This simple Sign Transformation can be proved as follows: Let the numbers to be multiplied be signed two's complement numbers x and y, viz.,

$$x = -x^0 + \sum_{i=1}^{N-1} x^i . 2^{-i} \qquad (5.3.7)$$

$$y = -y^0 + \sum_{j=1}^{M-1} y^j.2^{-j} \qquad (5.3.8)$$

Therefore the product $xy = (-x^0 + \sum_{i=1}^{N-1} x^i.2^{-i})(-y^0 + \sum_{j=1}^{M-1} y^j.2^{-j})$

$$= x^0.y^0 -y^0(\sum_{i=1}^{N-1} x^i.2^{-i}) -x^0(\sum_{j=1}^{M-1} y^j.2^{-j})$$

$$+(\sum_{i=1}^{N-1} x^i.2^{-i})(\sum_{j=1}^{M-1} y^j.2^{-j}) \qquad (5.3.9)$$

where the bits in the middle two summation terms are negatively weighted. Now, for any positive binary number, its negative value in two's complement notation is given by the following identity:

$$-\sum_{k=1}^{R-1} z^k.2^{-k} = -2^0 +(\sum_{k=1}^{R-1} \bar{z}^k.2^{-k} +2^{-(R-1)}) \qquad (5.3.10)$$

where $\bar{z}^k$ denotes the inverted or negation value of z^k. Therefore, for any bit

$$z^k = 1 -\bar{z}^k \qquad (5.3.11)$$

Substituting (5.3.11) in (5.3.9), we have

$$xy = x^0.y^0 -y^0(\sum_{i=1}^{N-1}(1-\bar{x}^i)2^{-i}) -x^0(\sum_{j=1}^{M-1}(1-\bar{y}^j)2^{-j})$$

$$+(\sum_{i=1}^{N-1} x^i.2^{-i})(\sum_{j=1}^{M-1} y^j.2^{-j})$$

$$= x^0.y^0 -y^0(\sum_{i=1}^{N-1} 2^{-i} - \sum_{i=1}^{N-1} \bar{x}^i.2^{-i}) -x^0(\sum_{j=1}^{M-1} 2^{-j} - \sum_{j=1}^{M-1} \bar{y}^j.2^{-j})$$

$$+(\sum_{i=1}^{N-1} x^i.2^{-i})(\sum_{j=1}^{M-1} y^j.2^{-j})$$

$$= x^0.y^0 -y^0(2^0 -2^{-(N-1)} -\sum_{i=1}^{N-1} \bar{x}^i.2^{-i}) -x^0(2^0 -2^{-(M-1)} -\sum_{j=1}^{M-1} \bar{y}^j.2^{-j})$$

$$+(\sum_{i=1}^{N-1} x^i.2^{-i})(\sum_{j=1}^{M-1} y^j.2^{-j})$$

Rearranging the middle two terms we have,

$$xy = x^{o}.y^{o} + y^{o}(-2^{o} + (\sum_{i=1}^{N-1} \bar{x}^{i}.2^{-i} + 2^{-(N-1)})) + x^{o}(-2^{o} + (\sum_{j=1}^{M-1} \bar{y}^{j}.2^{-j} + 2^{-(M-1)})) + (\sum_{i=1}^{N-1} x^{i}.2^{-i})(\sum_{j=1}^{M-1} y^{j}.2^{-j}) \quad (5.3.12)$$

Hence, by equation (5.3.10), the two middle terms in equation (5.3.12) represent the two's complements of the two middle terms in equation (5.3.9). Therefore, by comparing equations (5.3.9) and (5.3.12), the validity of the Sign Transformation is proved.

The above Sign Transformation can also be modified by rewriting equation (5.3.12) as follows:

$$xy = x^{o}.y^{o} + y^{o}(-2^{o} + \sum_{i=1}^{N-1} \bar{x}^{i}.2^{-i}) + y^{o}.2^{-(N-1)} + x^{o}(-2^{o} + \sum_{j=1}^{M-1} \bar{y}^{j}.2^{-j}) + x^{o}.2^{-(M-1)} + (\sum_{i=1}^{N-1} x^{i}.2^{-i})(\sum_{j=1}^{M-1} y^{j}.2^{-j}) \quad (5.3.13)$$

where the two middle summation terms now represent thw one's complements of the two middle summation terms in equation (5.3.9). This implies that step two of the previous Sign Transformation is now modified to become the one's complementation of the two words in the rectangles instead. However, in step three, the summation of all partial products now includes the sum of two extra weighted bits, viz., $y^{o}.2^{-(N-1)}$ and $x^{o}.2^{-(M-1)}$.

The procedure of the Modified Sign Transformation can be summarized as follows:

STEP 1: Reshuffle the bit pattern as in the Sign Transformation previously described.

1	0	0	1
1	0	0	1
1	0	0	1
1	0	0	1

STEP 2: One's complement the two words in rectangles. Note that the bit inside the square is common to both words, and always remains unchanged, as in the case of the Sign Transformation under two successive complementations.

0	0	0	1
0	0	0	1
0	0	0	1
1	1	1	0

STEP 3: Apply a "shearing" to the above data block so as to restore the previous relative weightings of the partial products and perform a straightforward binary summation of all partial products plus the terms $y^o.2^{-(N-1)}$ and $x^o.2^{-(M-1)}$.

```
                    0 0 0 1
                  0 0 0 1
                0 0 0 1
              1 1 1 0
                    1                 y^o.2^-(N-1)
                  + 1                           x^o.2^-(M-1)
IGNORE END-CARRY  (1)0.0 0 0 1 1 1    y_n
```

Before introducing the CBFCS algorithm, we first consider a hardware implementation of the above Modified Sign Transformation applied to the computation of y_n as given by equation (5.2.7) in the previous section. Let the filter variables and coefficients be represented by four-bit two's complement numbers, that is M=N=4. For a second-order difference equation such as equation (5.2.1), the partial products of y_n can be realized with memory hardware which is addressed by the filter variable bits

$$(x_n^j,\ x_{n-1}^j,\ x_{n-2}^j,\ y_{n-1}^j,\ y_{n-2}^j)$$

In the high-speed parallel implementation, we call for a total of four such memories, each containing the same content, that is all possible combinations of the partial products of y_n computed as in equation (5.2.7). Therefore, the size of each memory is 2^5x4 and these memories are simultaneously addressed by all filter variable bits as shown in figure 5.3.4. The outputs from these memories have to be "sign-transformed" before they can be summed to obtain the final product y_n as previously explained. However, if the Modified Sign Transformation is used, its multi-step procedure can be predone within the memories themselves by first transforming

the partial products of y_n prior to storing them in the memories. In this case, equation (5.3.13) is used.

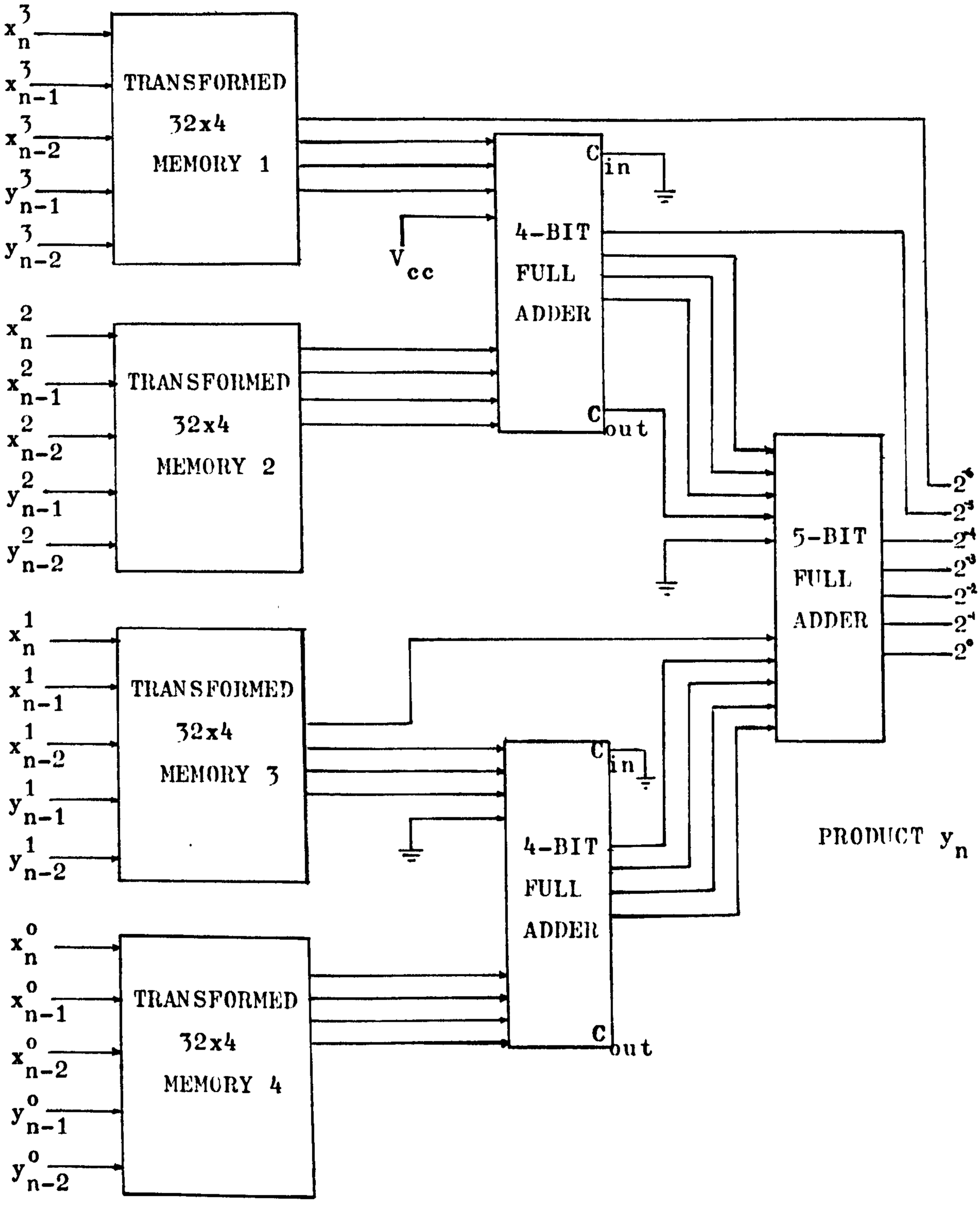


FIG: 5.3.4. USE OF A "MODIFIED SIGN TRANSFORMATION" IN A HIGH-SPEED PARALLEL HARDWARE IMPLEMENTATION.

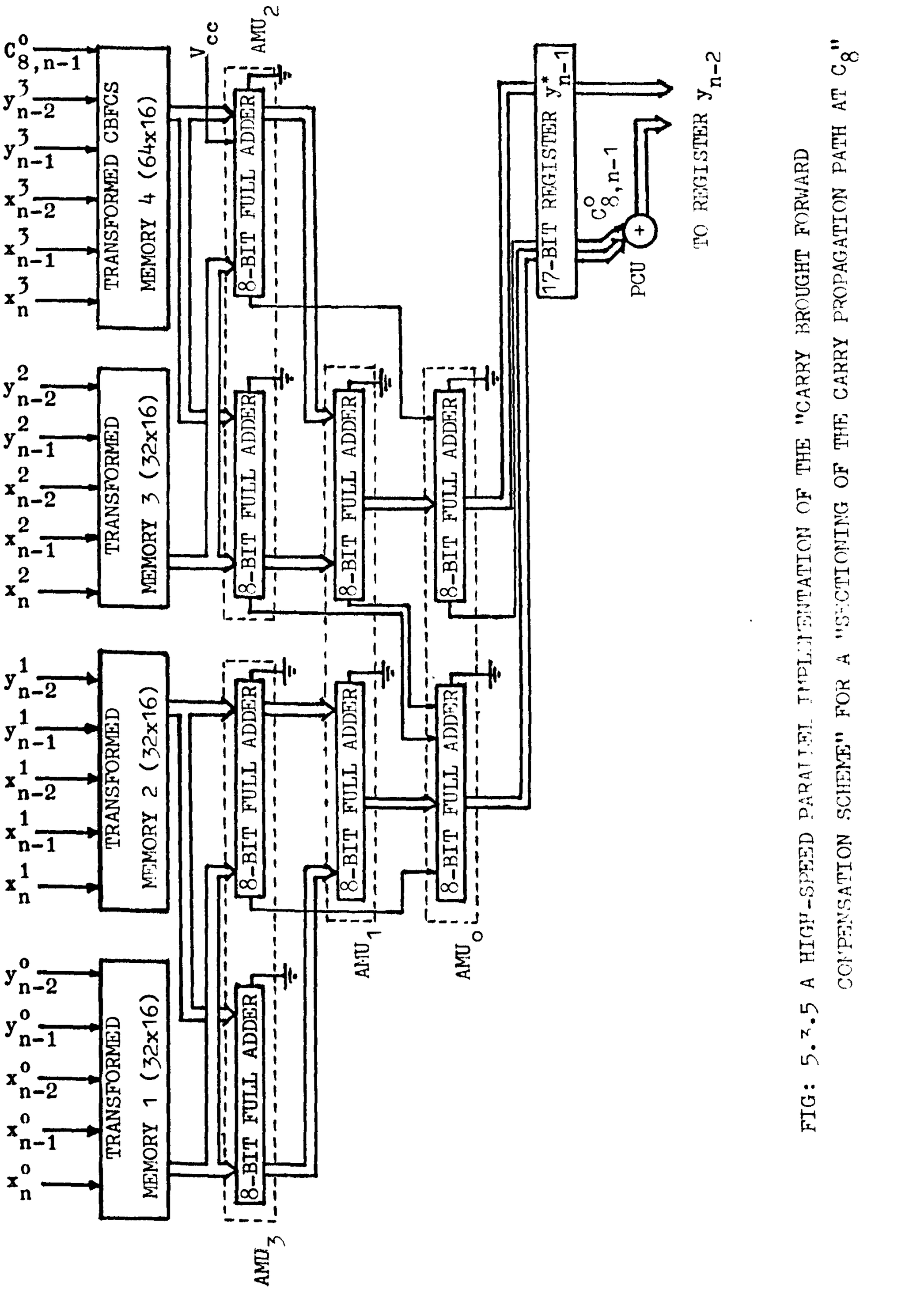
The two one's complementations performed in step two of the Modified Sign Transformation can be simply effected by inverting all sign bits of the partial products before storing them in the memories 1, 2 and 3 in figure 5.3.4, addressed by the 1st, 2nd and 3rd bits of the filter variables. The memory 4, which is addressed by the sign bits of the filter variables, will store the one's complements of all partial products with their sign bits inverted. The terms $y^{o}.2^{-(N-1)}$ and $x^{o}.2^{-(M-1)}$ can either be absorbed in the memory hardware or hard-wired to the 4-bit adders with the proper right shifts due to the factors $2^{-(N-1)}$ and $2^{-(M-1)}$ as shown in figure 5.3.4 as a logical 1 (i.e. V_{cc}).

In the above example, N=M=4 and only three standard 4-bit full adders are used. However, as the wordlength N of the partial products of y_n becomes longer, there will be an invariably longer carry propagation path in the summation of the partial products which governs the operational speed of the digital filter. Having dealt with the Modified Sign Transformation, we are now in a position to introduce the proposed CBFCS algorithm in the parallel high-speed implementation above for N larger than four.

Consider the high-speed parallel implementation of the CBFCS algorithm for a general second-order difference equation given by equation (5.2.1). Assume that N=16 bits, including sign, are used to represent the filter coefficients in two's complement notation. Let the carry propagation path be sectioned at the point $C_{B=8}$ for each summation AMU used. Furthermore, if the filter variables are represented by M=4 bits, including sign, in two's complement notation, then figure 5.3.5 depicts the high-speed parallel CBFCS implementation for the above example. The operation of the parallel implementation of the CBFCS algorithm can be explained by first re-writing equation (5.2.1) as follows:

$$y_n = a_o.x_n + a_1.x_{n-1} + a_2.x_{n-2} - b_1\left(y^{*}_{n-1} + C^{o}_{B,n-1}.2^{-(B-1)} + \ldots \right.$$

$$\left. \ldots + C^{o}_{\left(\frac{N}{B}-1\right)B,n-1}.2^{-\left(\left(\frac{N}{B}-1\right)B-1\right)}\right) - b_2.y_{n-2} \qquad (5.3.14)$$



5.7.5 A HIGH-SPEED PARALLEL IMPLEMENTATION OF THE "CARRY BROUGHT FORWARD COMPENSATION SCHEME" FOR A "SECTIONING OF THE CARRY PROPAGATION PATH AT C_8"

for the "sectioning" of the carry propagation path into (N/B) bytes at the "sectioning points"

$$(C_B, \ldots\ldots, C_{\left(\frac{N}{B}-1\right)B})$$

and the set of "brought forward" carry-outs

$$\{C^o_{B,n-1}, \ldots\ldots\ldots, C^o_{\left(\frac{N}{B}-1\right)B,n-1}\}$$

is the delayed version of the set of carry outputs of AMU_o in the computation of the nth output sample y_n. Also y^*_{n-1} is a "depleted" form of y_{n-1} in the parallel "byte-sliced" summation of the partial products of y_n, such that

$$y_{n-1} = y^*_{n-1} + C^o_{B,n-1}.2^{-(B-1)} + \ldots. + C^o_{\left(\frac{N}{B}-1\right)B,n-1}.2^{-\left(\left(\frac{N}{B}-1\right)B-1\right)} \qquad (5.3.15)$$

Since we only "sectioned" the carry propagation path at one point, $C_{B=8}$, in the above example, therefore, equation (5.3.14) becomes

$$y_n = a_o.x_n + a_1.x_{n-1} + a_2.x_{n-2} - b_1(y^*_{n-1} + C^o_{8,n-1}.2^{-7}) - b_2.y_{n-2} \qquad (5.3.16)$$

In order to calculate the memory requirement of the high-speed parallel implementation of the CBFCS algorithm, we define the following functions:

$$F(x^j_n, x^j_{n-1}, x^j_{n-2}, y^{*j}_{n-1}, y^j_{n-2})$$

$$= a_o.x^j_n + a_1.x^j_{n-1} + a_2.x^j_{n-2} - b_1.y^{*j}_{n-1} - b_2.y^j_{n-2} \qquad (5.3.17)$$

for the memories $j \neq M$, as shown if figure 5.3.5, and

$$F'(x^j_n, x^j_{n-1}, x^j_{n-2}, y^{*j}_{n-1}, y^j_{n-2}, C^o_{8,n-1})$$

$$= a_o.x^j_n + a_1.x^j_{n-1} + a_2.x^j_{n-2} - b_1.y^{*j}_{n-1} - b_2.y^j_{n-2} - b_1.2^{-(B-M)}.C^o_{8,n-1} \qquad (5.3.18)$$

for the CBFCS memory M in the above example.

Substituting equations (5.3.17) and (5.3.18) in (5.3.14), the final product of the above example can now be written in terms

of its partial products as

$$y_n = 2^{-(M-1)}.F'(x_n^{M-1},\ x_{n-1}^{M-1},\ x_{n-2}^{M-1},\ y_{n-1}^{*M-1},\ y_{n-2}^{M-1},\ C_{8,n-1}^{o})$$

$$+\sum_{j=o}^{M-2} 2^{-j}.F(x_n^j,\ x_{n-1}^j,\ x_{n-2}^j,\ y_{n-1}^{*j},\ y_{n-2}^j) \qquad (5.3.19)$$

for M=4, and the "byte-sliced" summation of the above partial products of y_n is performed as shown in figure 5.3.5.

In the above "byte-sliced" summation process, a number of carry-outs will be produced from AMU_1, AMU_2 and AMU_3 as depicted in figure 5.3.5. These carry-outs, while properly weighted with their respective right-shifts, will then be "brought forward" to, and "compensated" by AMU_o. Now the output of AMU_o consists of a "depleted" form of y_n and a carry-out $C_{8,n}^{o}$ which are connected to the register y_{n-1}^{*}. The above "depleted" form of y_n is finally "compensated" at the output of the register y_{n-1}^{*} by the "Product Compensation Unit" (PCU) via the following relationship:

$$y_{n-1} = y_{n-1}^{*} + C_{8,n-1}^{o} \qquad (5.3.20)$$

The individual bits of y_{n-1} and $C_{8,n-1}^{o}$ are used to address the M memories as shown in figure 5.3.5 for the computation of the next output sample of the filter.

In addition to the above mentioned, it is assumed that all partial products calculated by equations (5.3.17) and (5.3.18) are first transformed by the "Modified Sign Transformation" prior to storing in the above M memories so that the outputs of these memories can be summed as if they were pure (unsigned) binary numbers. From equation (5.3.17), the function F is seen to be a function of five variables and can therefore be realized by a memory with $2^5=32$ locations storing all possible combinations of the partial products

$$a_o.x_n^j + a_1.x_{n-1}^j + a_2.x_{n-2}^j - b_1.y_{n-1}^{*j} - b_2.y_{n-2}^j$$

On the other hand, the function F' in equation (5.3.18) is a function

of six variables realized by a CBFCS memory with $2^6=64$ locations containing all possible combinations of the CBFCS partial products

$$a_o.x_n^j + a_1.x_{n-1}^j + a_2.x_{n-2}^j - b_1.y_{n-1}^{*j} - b_2.y_{n-2}^j - b_1.2^{-(B-M)}.C_{8,n-1}^o$$

for j=0 to j=M-1. It is assumed that B>M and the CBFCS memory will have its first 32 locations same as the other memories defined by the function F and the last 32 locations as the individual contents of the first 32 locations incremented each by the value

$$b_1.2^{-(B-M)} \qquad (5.3.21)$$

In case of B<M, the weighting in equation (5.3.21) will be larger than unity and a memory whose output is weighted by the factor 2^{-j}, where j=0 to j=B-1, will be used as a CBFCS memory instead. In this case, equation (5.3.21) is modified to

$$b_1.2^{-(B-j-1)} \qquad (5.3.22)$$

as the outputs of the M memories are respectively weighted by a factor 2^{-j} where j=0 to j=M-1. In particular, when M=B, then the last 32 locations of the CBFCS memory M will contain the individual contents of the first 32 locations each incremented simply by the value b_1. This further implies that the last 32 locations can be realized by a logical left-shift of the first 32 locations and the size of the CBFCS memory M can be correspondingly reduced to only 32 locations instead of the original 64. A reduction in CBFCS memory size from 64 down to 48 is also possible for B>M with a suitably decoded address system for the variables

$$(x_n^{M-1},\ x_{n-1}^{M-1},\ x_{n-2}^{M-1},\ y_{n-1}^{*M-1},\ y_{n-2}^{M-1},\ C_{8,n-1}^o)$$

Nevertheless, equation (5.3.18) gives the "Carry Brought Forward Compensation" for the high-speed parallel implementation of the CBFCS algorithm modified by the "Modified Sign Transformation". Its is defined in a manner similar to equation (5.2.10) for the function f in the serial implementation of the CBFCS algorithm.

In short, there are in general M AMU's and M memories as required in the high-speed parallel implementation of the CBFCS algorithm; hence the tradeoff between hardware and speed. All of the M memories are simultaneously addressed and their outputs (the partial produsts) are summed in a "byte-sliced" manner, according to their respective weightings (that is the factor 2^{-j}). The carry-outs "brought forward" from AMU_1, ..., AMU_{M-1} to AMU_o are first compensated by the unit which is thus functionally equivalent to the "Compensation Unit" CU' in the serial implementation. The carry-outs from AMU_o is used to compensate its "depleted" output sample at a later stage via the "Product Compensation Unit" (PCU) which is thus functionally equivalent to the "Compensation Unit" CU" in the serial implementation. The summation in equation (5.3.20) is performed with an 8-bit full adder in some "dead time" of the system prior to loading the "compensated product" y_{n-1} to the register (not shown in figure 5.3.5) for the output sample y_{n-2}. The above 8-bit full adders that make up the AMU's are each made up of two standard 4-bit full adders with ripple carry propagation between them. As in the previous serial implementation, these 4-bit adders are regarded as hard-wired logic function units in the "sectioning" of the N-bit worst case carry propagation path. Besides, the arrangement of the AMU's in figure 5.3.5 is only one of the many possible combinations, and, in general, an arrangement should be found such that a minimum number of standard 4-bit adders are required.

We shall now examine the speed of operation of this parallel implementation of the proposed CBFCS algorithm modified by the "Modified Sign Transformation". To begin with, we consider the realization of recursive difference equations as in figure 5.3.5. Essentially two operations have to be completed before a new data word can be shifted into the system. These operations are:

(1) Evaluations of the functions F and F'. The time taken is t_m which is the memory access time.

(2) "Byte-sliced" summation of all partial products simultaneously. The time required is the total time in going through the $(1+\log_2 M)$ layers of AMU's, with t_a as the summation time of an AMU.

Consequently, the word rate for the high-speed parallel implementation of recursive difference equations is:

for $\log_2 M$ an integer,
$$\frac{1}{t_m + t_a(1+\log_2 M)} \tag{5.3.23}$$

For non-integral value of $\log_2 M$, we have

$$\frac{1}{t_m + t_a(1+\log_2 M)} \tag{5.3.24}$$

where (x) means the smallest integer greater than x.

Finally, if we introduce a pair of registers for each AMU required in the hardware implementation of a non-recursive difference equation, the word rate of the high-speed parallel implementation becomes

$$\min\left\{\frac{1}{t_m}, \frac{1}{t_a}\right\} \tag{5.3.25}$$

which is obviously much faster than the case for recursive difference equations. The introduction of a pair of registers for each AMU has turned the implementation of non-recursive difference equations into a pipeline system which accounts for the increase of speed.

To conclude this section, we point out that so far we have described two extreme modes of implementation of the proposed CBFCS algorithm which greatly exploits memory hardware to perform addition and multiplication. In short, the high-speed parallel mode offers an increase in speed at the expense of more hardware, while the comparatively slow-speed serial mode requires the minimum amount of memory hardware. In addition, the complexity of the control circuits is maximum for the serial mode and minimum for the parallel mode. However, on the other hand, power consumption is minimum for the serial mode while maximum for the parallel mode. Furthermore, a compromise between the serial and parallel modes is always feasible, yielding a "medium-speed" hybrid mode of implementation. The exact mode of implementation to be used depends on the demands of individual circumstances. The advantage of the serial mode is economy in hardware and a tradeoff between accuracy (wordlength N)

and operational speed. The advantage of the parallel mode is maximum operational speed and a tradeoff between accuracy and hardware. In general, a tradeoff exists between the amount of hardware required and the resultant operational speed for a given accuracy. In the next section, we shall describe the implementations of high-order and multiplexed digital filters via the proposed CBFCS approach.

5.4 IMPLEMENTATIONS OF HIGH-ORDER AND MULTIPLEXED DIGITAL FILTERS:

As mentioned previously in section 4.1, one almost always implements an Nth order digital filter as a cascade or parallel combination of first- and second-order digital filters due to quantization noise effect. In this section, we describe the general approach to the realization of high-order digital filters and multiplexed digital filters using the serial-mode second-order section described in section 5.3.2 as a building block. For maximum operational speed, these serial-mode second-order sections are then replaced by the high-speed parallel-mode second-order sections described in section 5.3.3.

When the signal being processed is not a wide-band one in comparison with the operational speed of the basic second-order section, the arithmetic units of the above second-order sections can be multiplexed to implement the Nth order digital filter more efficiently and cost-effectively. The various multiplexing schemes are of two main types: (1) the multiplexed filter may operate upon a number of input signals simultaneously if the individual input rate is significantly below its capability, or (2) the multiplexed filter may effect a number of different filters for a single input signal. In the former case, the digital filter is acting as a "fixed" filter without changing its coefficients. Thus to multiplex the digital filter to process M simultaneous inputs, the input samples from the M channels are interleaved sample by sample and fed serially into the filter. The output samples emerge in the same interleaved order as the input and are thus easily seperated into M different output channels. However, in the latter case, the digital filter acts as a variable-coefficient filter and is of more importance than the former straightforward case. We therefore concentrate on the design of the latter case in this section.

Since the proposed CBFCS approach described in the previous section leads to high-speed hardware implementations of realtime digital filters, the basic second-order section will be multiplexed in the implementations of high-order and multiplexed filters. Consider a 4th order digital filter realized in cascade form as shown in figure 5.4.2. The block diagram of the 4th order digital filter, with transfer function H(z), is shown in figure 5.4.1 below.

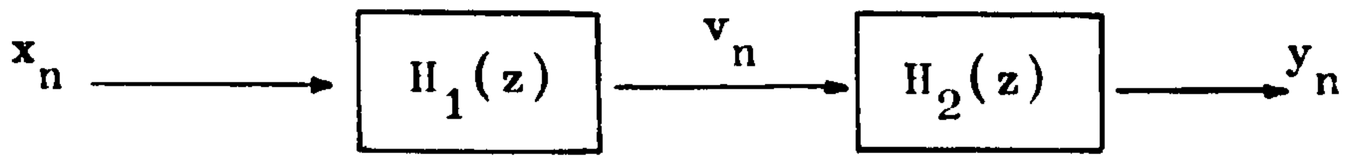


FIG: 5.4.1 A 4TH ORDER DIGITAL FILTER
REALIZED IN CASCADE FORM

In figure 5.4.1,

$$H(z) = H_1(z).H_2(z)$$

where $H_1(z)$ and $H_2(z)$ are basic second-order sections.

In figure 5.4.2, the necessary multiplexer and corresponding demultiplexer at the input and output ends respectively are standard integrated circuits which are available as off-the-shelf items. The CBFCS memory in the above cascade realization contains the "compensated partial products" of the transfer functions $H_1(z)$ and $H_2(z)$ in two seperate sections which are selected by the "section selection" control. The "selection control" is a direct extension of the CBFCS address system. In practice, it can be simply furnished by a divide-by-two counter which is only a JK flip-flop.

Consider next, a multiplexed 4th order digital filter implemented in parallel form as shown in figure 5.4.3. The block diagram of the 4th order digital filter transfer function H(z) is shown in figure 5.4.4. In figure 5.4.3, it is noticed that the hardware implementation of a multiplexed digital filter in parallel form differs from that of a cascade form in the input register and the output adder. The input register for the sample x_n is wired as shown in figure 5.4.3 with its output fed back twice to its input

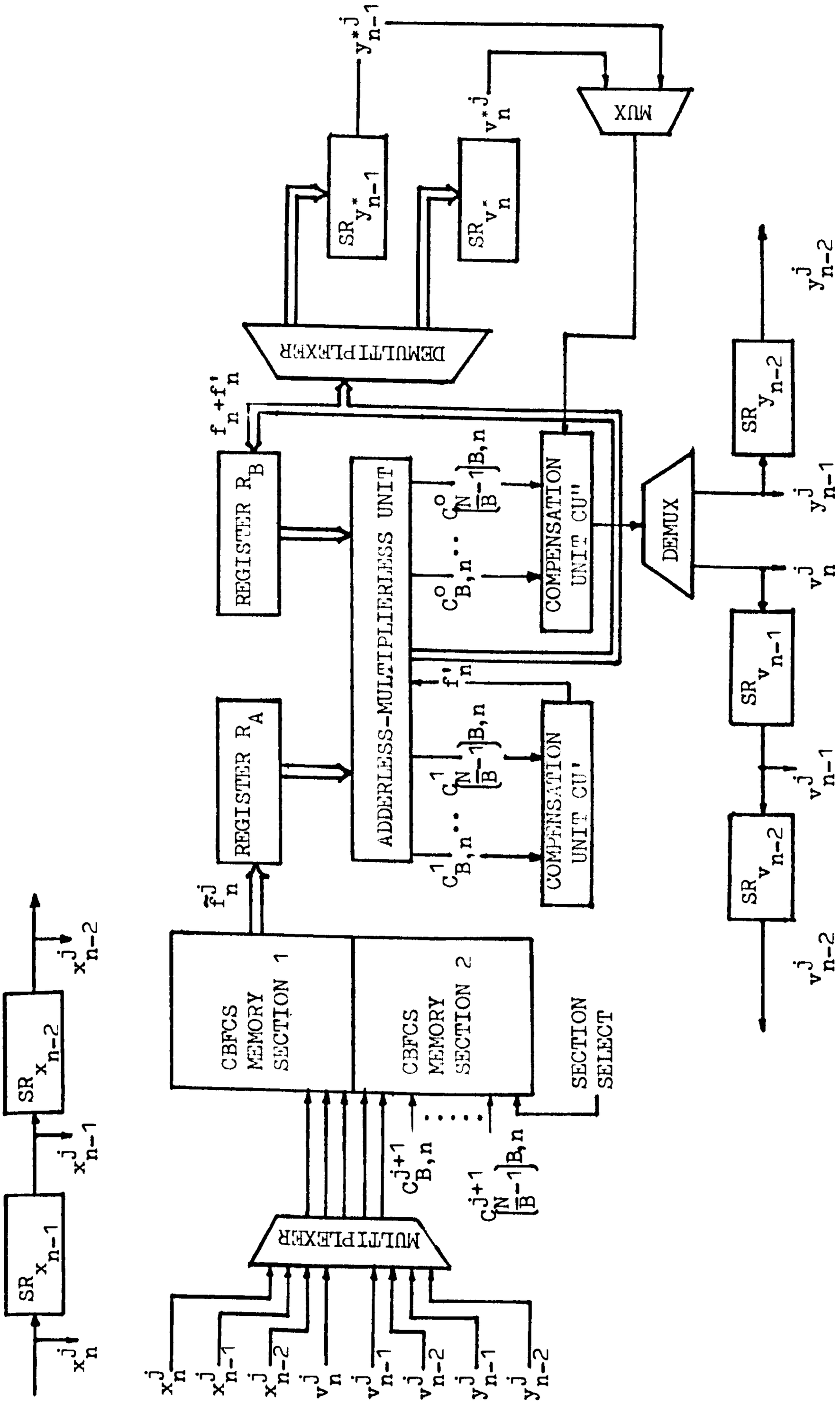


FIG: 5.4.2 A MULTIPLEXED 4TH ORDER DIGITAL FILTER IMPLEMENTED IN THE CASCADE FORM

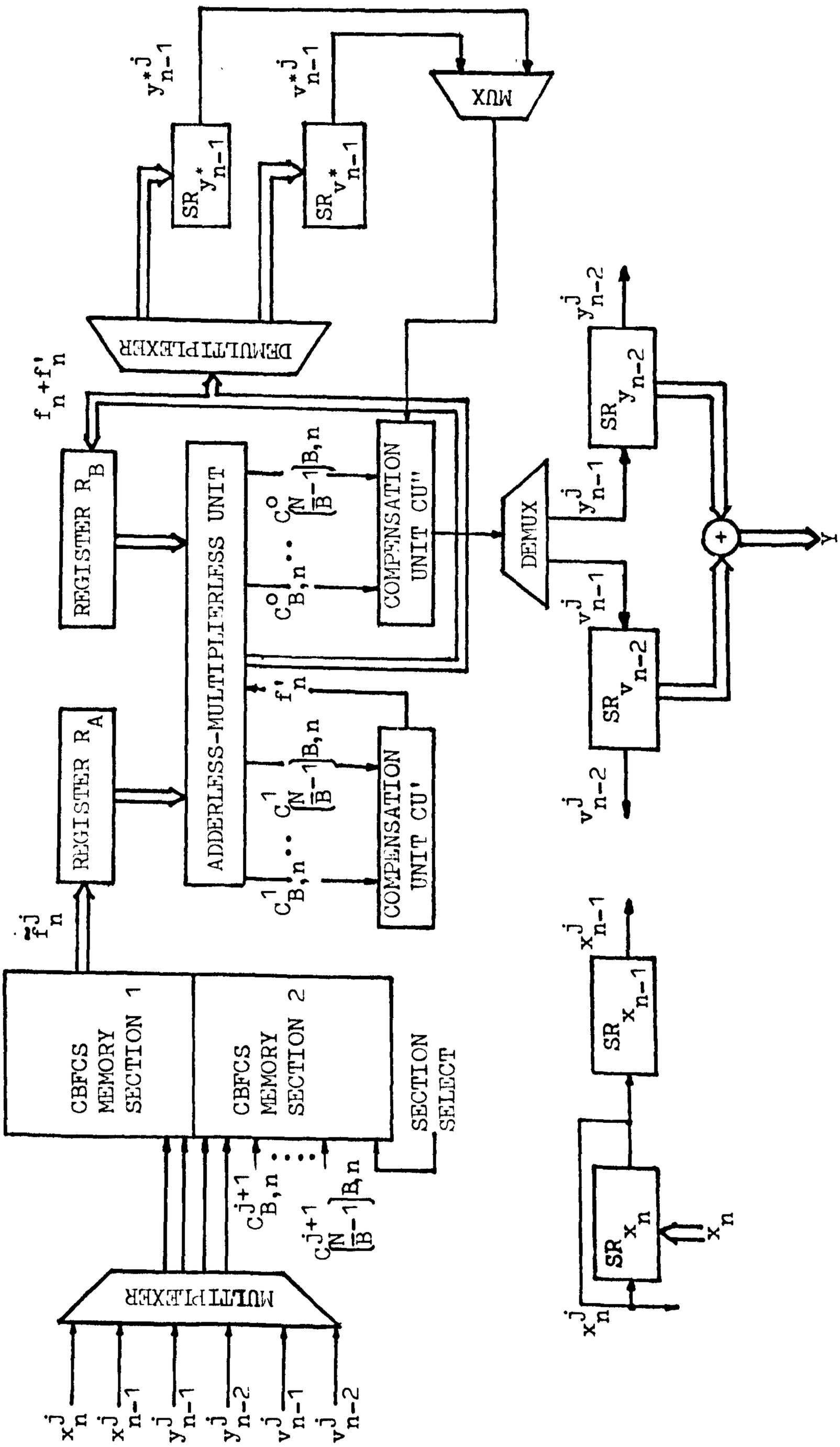


FIG: 5.4.3 A MULTIPLEXED 4TH ORDER DIGITAL FILTER IMPLEMENTED IN THE PARALLEL FORM

for the two parallel sections $H_1(z)$ and $H_2(z)$. The input sample x_n is loaded every so often in parallel into the register when its content has been circulated twice around. In practice, it can also be realized with a sequential access memory (SAM) which is a shift register that can circulate its content every so often as required via a recirculate control on the integrated circuit itself. The output summer is in fact realized in practice with a low-speed parallel input adder, as this summation can be performed in a relatively long time compared with the summation time of the high-speed AMU (a ratio of 1:M+1). As before, the "section select" input is addressed by a binary counter.

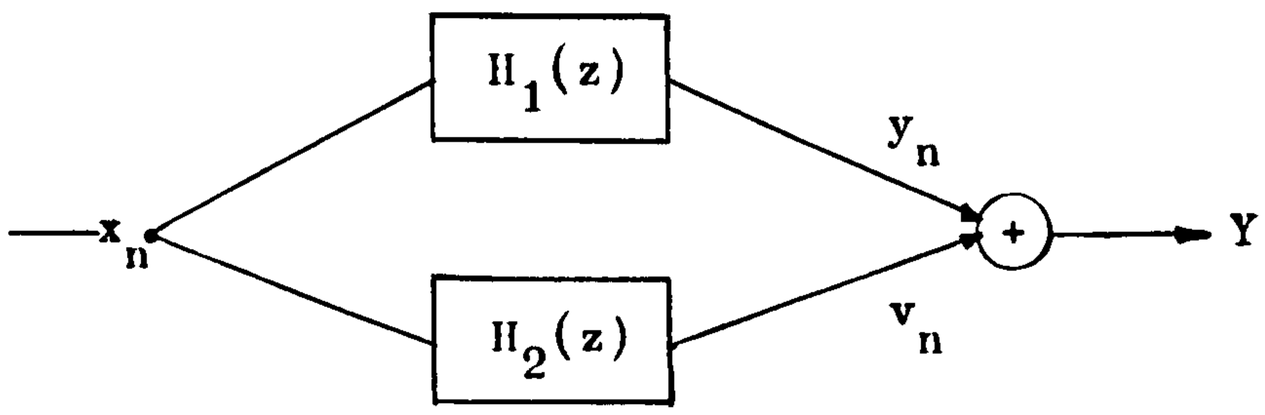


FIG: 5.4.4 A 4TH ORDER PARALLEL DIGITAL FILTER BLOCK DIAGRAM

The above parallel form, multiplexed high-order digital filter implementation is particularly suitable for filter bank realizations. In fact, it can be directly used to implement the proposed design technique for high-speed realtime digital filters. For example, equation (3.2.28) in section 3.2.6 can be realized as a parallel combination of $(M_l + M_u + N)$ second-order sections. If the input data rate is low when compared with the operational speed of the AMU, then the basic second-order section can be multiplexed in a manner as shown in figure 5.4.3. In this instance, the CBFCS memory is enlarged to $(M_l + M_u + N)$ sections containing all the "compensated partial products" of the individual second-order sections. The input to the "section select" will then be addressed by a divide-by-$(M_l + M_u + N)$ binary counter.

To conclude this section, we now suggest an alternative second-order hardware implementation of the individual sections of

the zero-sharing structure given by equation (3.3.3) in section 3.3. This follows from a direct simplification of the serial-mode second-order implementation shown in figure 5.3.3. A second-order section in the above zero-sharing structure has the following difference equation:

$$y_n = x_n - x_{n-1} - b_1 \cdot y_{n-1} - b_2 \cdot y_{n-2} \qquad (5.4.1)$$

Now the input terms in equation (5.4.1) above can be rewritten as the differential input

$$\Delta x = x_n - x_{n-1} \qquad (5.4.2)$$

where Δx can simply be obtained by a two's complement subtraction of the input samples x_n and x_{n-1}. The resultant difference Δx is now fed into the "modified" serial-mode digital filter as shown in figure 5.4.5.

The input sample Δx is initially loaded in parallel into the register R_B as shown in figure 5.4.5, through the multiplexer wired in parallel to its input. The circuit operates in the same serial manner as in figure 5.3.3 and equation (5.2.11) is now modified to become

$$f_n = -f(y^0_{n-1},\ y^0_{n-2},\ C^1_{B,n}, \ldots\ldots,\ C^1_{\left[\frac{N}{B}-1\right]B,n})$$

$$\sum_{j=1}^{M-1} 2^{-j} \cdot f(y^j_{n-1},\ y^j_{n-2},\ C^{j+1}_{B,n}, \ldots\ldots\ldots\ldots,\ C^{j+1}_{\left[\frac{N}{B}-1\right]B,n}) \qquad (5.4.3)$$

and as given by equation (5.2.14)

$$y_n = f_n + f'_n + f''_n$$

where f'_n and f''_n are values of the compensation functions f' and f" respectively, as given by equations (5.2.12) and (5.2.13), in the computation of the nth output sample y_n.

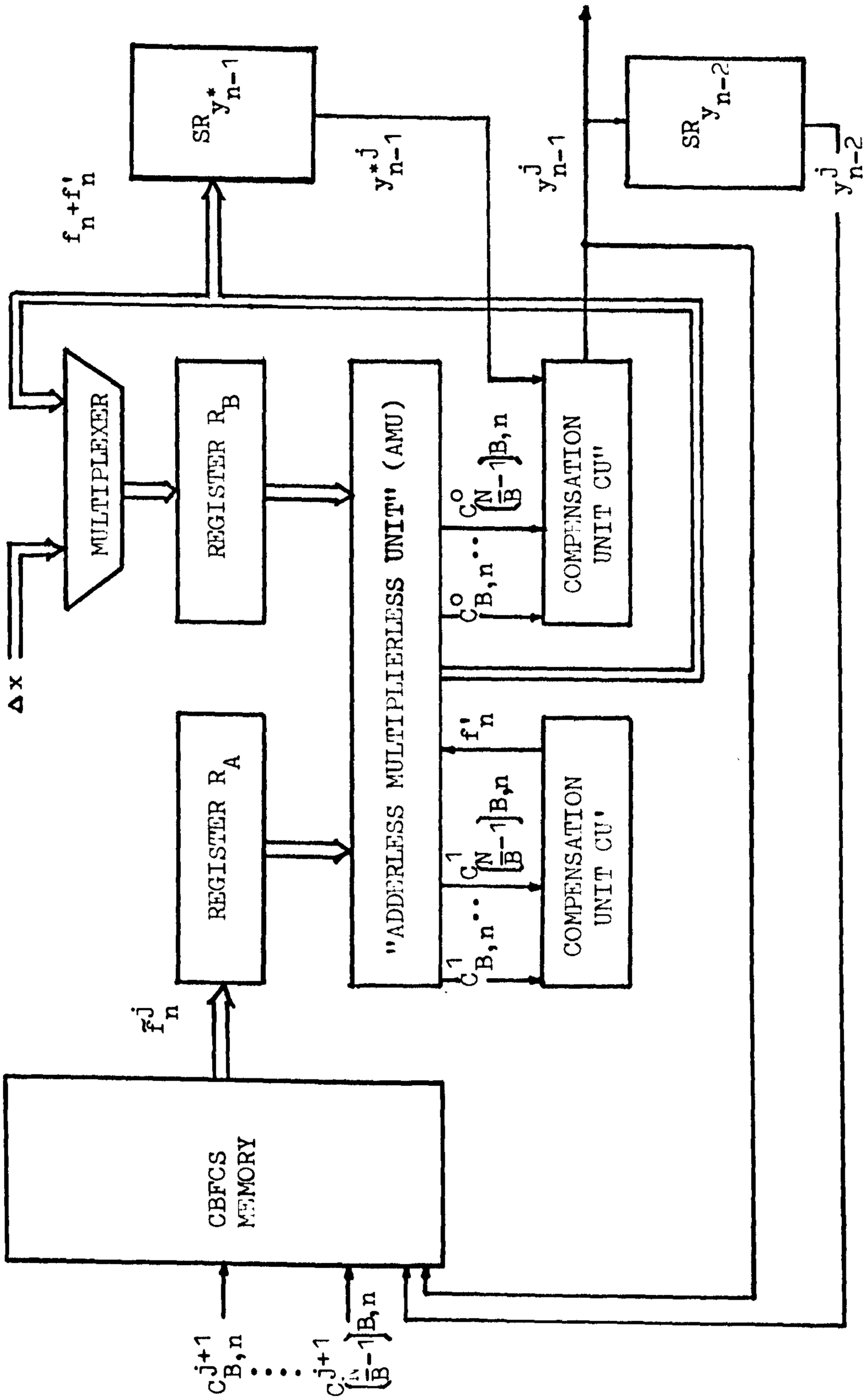


FIG: 5.4.5 A MODIFIED 2ND ORDER SERIAL IMPLEMENTATION FOR A ZERO-SHARING FILTER

The number of binary arguments of the function f in equation (5.4.3) is now reduced by three as compared with equation (5.2.10) for a general second-order difference equation. This implies that the number of binary states (the total number of possible combinations of the compensated partial products of f_n) represented by equation (5.4.3) is now 8 (or 2^3) times less than that for a general second-order section as given by equation (5.3.1). Thus, an 8-fold saving in the CBFCS memory requirement is now feasible with the above modified structure for equation (5.4.1) as shown in figure 5.4.5.

Furthermore, if the above structure is to be multiplexed to implement the proposed design technique for a filter bank, then the coefficients of individual channels can be calculated via equation (3.6.4) in chapter three. Of course, random access memory (RAM) will be used for the CBFCS memory realization, as the basic second-order section is behaving as a variable filter in this context. This then implies a tremendous saving in CBFCS memory requirement for the whole filter bank, as the coefficients of individual filters do not have to be stored in a coefficient memory whose size may become excessive for a large filter bank. Equation (3.6.4) in section 3.6 provides an efficient means of generating a set of coefficients of one elemental filter from another adjacent elemental filter in the bank. From this interpolation formula, it is obvious that we only have to furnish the term cos(nbT) while the term cos(n+1)bT can then be given by the difference of the terms cos(nbT) and sin(nbT)sin(bT) which represents a small iterpolation value sin(bT), since the angle bT is small for a narrowband elemental filter. Hence, while the AMU of the basic second-order section is processing an input sample x_n for one elemental filter, the CBFCS memory content for an adjacent elemental filter can be computed and written into the above mentioned RAM. The process of realtime computation of the CBFCS memory content continues until the complete filter bank has been scanned across by the variable basic second-order section.

5.5 CONCLUSION AND A COMPARISON WITH EXISTING HARDWARE IMPLEMENTATIONS OF DIGITAL FILTERS:

In this chapter, we have presented a high-speed hardware implementation of digital filters via the CBFCS algorithm. The proposed CBFCS algorithm relies on the recent advances in semiconductor technology to yield high operational speeds and economy in hardware in realtime digital filter implementations. In this approach, the additions and multiplications required in an Nth order difference equation have been replaced with an "Adderless-Multiplierless Unit" (AMU) in conjunction with the CBFCS memory. Apparently, what remains next to be discussed in this chapter is a comparison of the proposed approach with other contemporary approaches in some "practical terms" viz., choice and cost of components vs speed and power consumption.

B.Liu et.al. in their recent paper (76) have proposed a "Combinatorial Filter" which relies on the bit-sliced technique described in section 4.3.2 in its implementation. Moreover, they have made a comparison of their approach with other existing approaches and their findings are such that, in general, the bit-sliced technique is by far more efficient than other methods in terms of speed, cost and power consumption. In our comparison that follows, we shall concentrate on comparing B. Liu et. al. approach with the proposed CBFCS approach.

To begin with, both approaches use memory in the hardware implementations of digital filters. However, B.Liu et.al. approach has only been able to replace multipliers with memory while the summation of partial products has been performed using conventional adders which suffer a fundamemtal speed limitation due to the N-bit worst case carry propagation path. In the proposed CBFCS algorithm the worst case carry propagation path has been "sectioned" down to a maximum of B bits where B is a submultiple of N, thereby overcoming the above speed limitation to some extent. We shall further base our comparison with B.Liu et.al. approach on a general second-order filter implementation in both cases with N=16 bits for the filter coefficients. In both serial implementations, the operational speed of the resultant filter is given by equations (4.3.23) and (5.3.4)

as $$\frac{1}{t_s} = \min\left\{\frac{1}{t_m}, \frac{1}{t_a}\right\}$$

where t_m = memory access time in both cases,
and t_a = addition time for two 16-bit numbers in parallel using conventional adders in B.Liu et.al. approach, and the AMU in the proposed approach.

Assuming that the same type of memory is used in bothe cases, then t_m will be the same for both cases. Consequently, our comparison will be restricted to the factor t_a in both approaches. In B.Liu et.al. approach, using standard TTL technology, a 16-bit conventional parallel full adder can be implemented with four 4-bit full adders, with full carry-lookahead within each adder and ripple carry propagation between adjacent adders. For such an arrangement of the four 4-bit adders, the addition time t_a is typically 43 nsec while the power comsumption is typically 1.24 watts. For the sake of cost comparison, we shall denote the cost of the above 16-bit conventional full adder as "x" units (price calculations is based on current manufacture retail figures).

Using the same four 4-bit standard TTL full adders, and assuming a carry propagation path sectioned at $C_{B=8}$ (that is a breaking of the path into two 8-bit bytes), the proposed CBFCS approach will give t_a as 23 nsec typically. However, we hasten to point out, the CBFCS requires more memory hardware than B.Liu et.al. approach. The cost of the maximum amount of extra memory in the CBFCS approach is 0.22x approximately, when using high-speed low-density Schottky memories, bringing the total cost of the CBFCS to 1.22x as compared with B.Liu et.al. approach. On the other hand, for this extra amount of memory, the proposed approach has achieved an operational speed nearly twice that of B.Liu et.al. approach. As regards power consumption, the CBFCS approach dissipates 0.4 watt marginally more than B.Liu et.al. approach, and this is due to the fact that semiconductor memory dissipates less when compared to the standard 4-bit adders. This fact also highlights the importance of of the CBFCS algorithm in replacing computational units with memory, as we shall see in the latter part of our comparison. Nevertheless,

the cost of the above extra memory required in the CBFCS approach will become substantially lower (0.001x) when lower speed high-density memory is used. The extra power consumption in this case is 0.035 watts more than B.Liu et.al. approach. However, the price paid for the reduced memory cost and power dissipation is a corresponding decrease in operational speed to 25MHz instead of 41MHz approximately in the previous case. However, this still compares marginally more favourable with B.Liu et.al. approach which when using the above 16-bit conventional adder, will only operate at 23MHz approximately. Nevertheless, it is seen that a tradeoff between addition time t_a and memory requirement exists, and this is of course an aim of the CBFCS approach.

Furthermore, in order to achieve the speed of the proposed CBFCS approach, B.liu et. al. approach has to call for some high-speed Schottky Arithmetic Logic Units (ALU) or the use of single-bit full adders. As the implementation with single-bit adders is more costly than the use of ALU's, we shall assume the latter instead. These ALU's are capable of performing the necessary additions at high-speed with the support of some Carry-lookahead Generators (CLAG) to provide full carry-lookahead across all sixteen bits. The cost of hardware implementation in this case is 10.7x and the power consumption is 2.7 watts approximately. Hence, B.Liu et.al. approach achieves the speed of the proposed approach at the expense of a higher cost and power consumption. This eventuality is, of course, due to the use of high-speed Schottky devices to increase the resultant speed performance. This further leads to the implication that B.Liu et.al. approach is dependent on the choice of technology for speed, while, on the other hand, given a particular technology, the CBFCS approach allows more flexibility and tradeoffs. For instance, if the above mentioned carry propagation path is further sectioned at the sectioning points (C_4, C_8, C_{12}), that is sectioning the path into bytes of B=4 bits, then the time t_a for the AMU, which is still made up of four 4-bit standard TTL full adders, will be 16nsec in the proposed CBFCS approach. This speed proves impossible for B.Liu et.al. approach as well as other contemporary approach in serial implementations of digital filters for a given hardware technology, e.g. TTL.

So far, our comparison has been based on the serial implementation. When the parallel implementation is used in the above example, the extra memory required in the proposed approach becomes negligible compared to the size of the original memory required. This, coupled with the resultant increase in speed due to parallel processing, therefore, compares even more favourably with B.Liu et.al. approach. In addition, as in the case of the serial implementation, if the carry propagation path of the above example is being sectioned at the sectioning points (C_4, C_8, C_{12}), then the resultant operational speed achieved by the use of the parallel implementation proves impossible for B.Liu et.al. approach as well as other contemporary parallel implementations for a given hardware technology, e.g. TTL.

In conclusion, the proposed approach does not include "adds" and "multiplies" of conventional arithmetic units as such, but instead, employs a novel CBFCS algorithm which brings together the recent advances of semiconductor memory and the effectiveness of standard 4-bit full adders wired as logic function units, to replace conventional computational units at a reduced cost, in the hardware implementation of high-speed realtime digital filters. As individual 4-bit full adders no longer behave as conventional adders in an N-bit parallel summation, but rather as specific combination logic elements, the worst-case N-bit carry propagation path of an N-bit parallel addition is now being sectioned down to a maximum of B bits long. Consequently, the proposed approach makes possible operational speeds which cannot be achieved by existing approaches in the implementation of digital filters. Furthermore, if ECL or other high-speed logic families are to be used in the proposed CBFCS approach, further speed improvements can be obtained. Nevertheless, B.Liu et.al. have claimed that their approach offers significant savings in terms of hardware cost and power consumption as well as speed improvement over other existing approaches. However, in the above comparison with B.Liu et.al. approach, the proposed CBFCS approach has clearly contributed a step further in hardware cost and power consumption reductions as well as speed improvements in high-speed filtering (in our comparison, the cost calculations have been based on $\chi = \chi_{max}$ whereas, in general, $\chi < \chi_{max}$ and more savings are then achievable).

Finally, we will like to point out that the discussions in the previous sections have not included the effects of finite word length. In the next chapter, the errors due to the use of finite word lengths in representing the filter coefficients and variables in the proposed CBFCS approach will be discussed and analysed in detail. Furthermore, in the discussion of the proposed CBFCS algorithm, the filter variables and coefficients have been assumed to be M-bit and N-bit fractions respectively, for the sake of convenience. In general, they can be mixed numbers (that is numbers with an integral part as well as a fractional part) and the CBFCS functions can be modified accordingly to accommodate mixed numbers.

CHAPTER SIX

ERROR ANALYSIS OF THE PROPOSED HARDWARE IMPLEMENTATION

6.1 INTRODUCTION:

When a digital filter is implemented on a computer or with special-purpose hardware, errors and constraints due to finite word length are inevitable. In the implementation of a linear time-invariant digital filter, the following sources of error arise from the use of finite word length:

(1) Input signal quantization.
(2) Filter coefficient quantization.
(3) Uncorrelated roundoff noise in multiplication.
(4) Correlated roundoff noise or limit cycles, including overflow oscillations.
(5) Dynamic range constraints such as the scaling of parameters to prevent overflows and underflows of the finite word length registers in a digital filter.

Furthermore, the choice of a particular type of arithmetic notation has often an important bearing on the above mentioned errors, and, in addition, if a digital filter is used to process analogue signals, there may be other sources of error as a result of sampling the analogue input signals and the reconstruction of output analogue signals, as pointed out in references (79)-(85).

In view of the above, we have devoted the present chapter to the analysis of finite word length effects on the proposed CBFCS algorithm in the hardware implementation of digital filters. Also, explicit expressions have been derived which allow the designer to choose the appropriate filter parameters to achieve a specified performance.

6.2 SIGNED FIXED-POINT REPRESENTATION OF PARAMETERS IN THE PROPOSED HARDWARE IMPLEMENTATION:

The coefficients and variables of a digital filter can be represented using a finite number of bits in either the fixed-point form or the floating-point form. In the proposed CBFCS algorithm

discussed in chapter five, we have used two's complement fixed-point notation to represent the filter coefficients and variables. Consequently, we shall concentrate on fixed-point arithmetic although floating-point arithmetic can also be used in the hardware implementation of the proposed CBFCS algorithm.

Moreover, the use of two's complement notation in the proposed approach throughout any additions and multiplications required, has an advantage in operational speed over systems with mixed notations. This is due to the fact that these mixed notation systems perform multiplications in sign-and-magnitude notation while additions are performed using two's complement notation, thus, suffering an inevitalbe loss in processing time whenever a change in notation is required in going from one arithmetic operation to another.

6.2.1 EFFECT OF TWO'S COMPLEMENT TRUNCATION:

Consider the general case where a fixed-point binary number is represented by (b+1) bits, including sign, as below:

$$x = -x^{o} + \sum_{i=1}^{b} x^{i}.2^{-i} \qquad (6.2.1)$$

The most significant bit to the left of the binary point is the sign bit which is given a negative weighting as shown in equation (6.2.1). There are b bits to the right of the binary point such that the range of numbers that can be accommodated is

$$-1 \leqslant x \leqslant (1 - 2^{-b}) \qquad (6.2.2)$$

and the magnitude of the least significant bit is 2^{-b} which is the width of quantization since the numbers are quantized in steps of 2^{-b}.

Truncation of a two's complement number is accomplished by simply discarding all bits less significant than the least significant bit that is retained. Thus if (b+1) bits are retained, the truncation error

$$E_T = Q[x] - x \qquad (6.2.3)$$

satisfies the inequality:

$$0 \geqslant E_T > -2^{-b} \qquad (6.2.4)$$

where x and Q(x) denote the two's complement number before and after truncation. The above nonlinear relationship representing the truncation of two's complement numbers is depicted in figure 6.2.1.

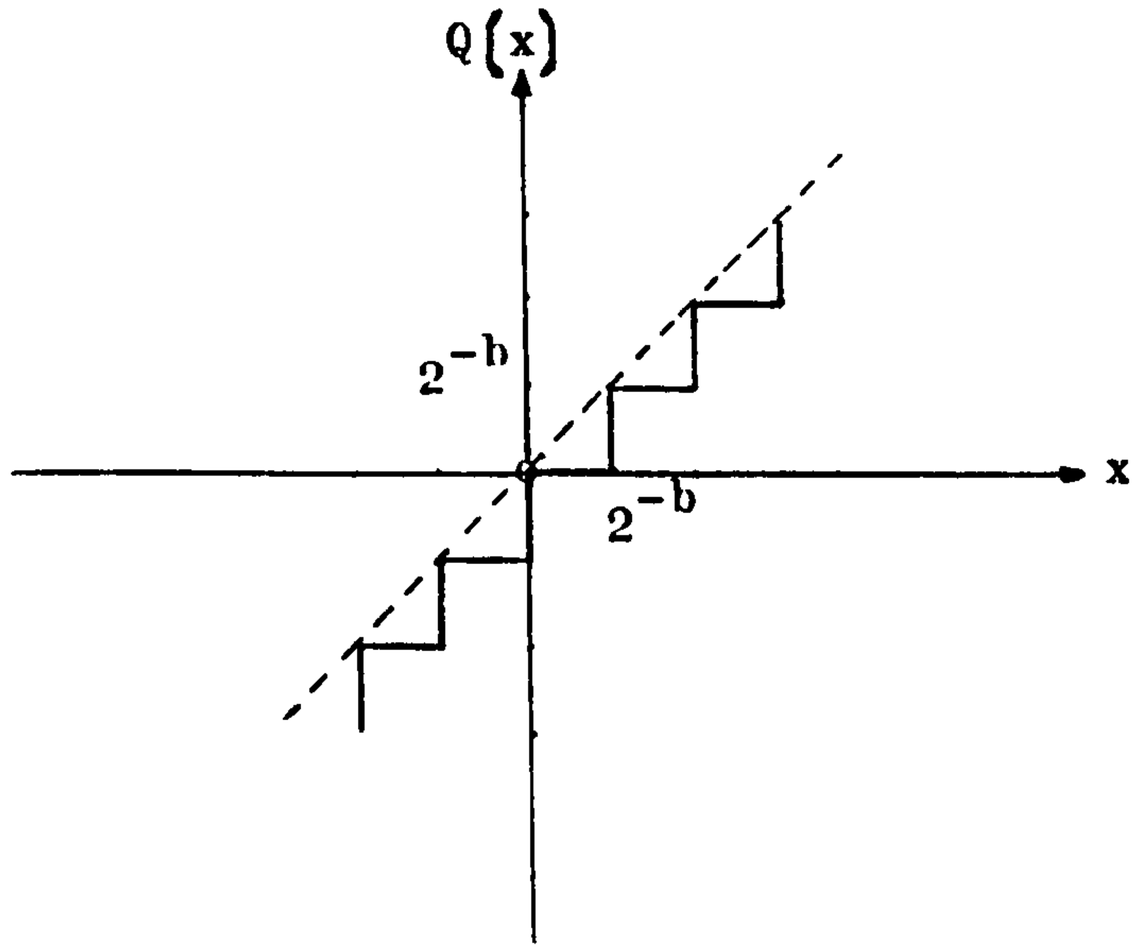


FIG: 6.2.1 NONLINEAR RELATIONSHIP REPRESENTING A TWO'S COMPLEMENT TRUNCATION QUANTIZER.

Furthermore, if we make the following assumptions: (86)-(91)

(1) The sequence of error samples $\{E_T\}$ is a sample sequence of a stationary random process.

(2) The random variables of the error process are uncorrelated, that is, the error process is a white-noise process.

(3) The probability distribution of the error process is uniform over the entire range of the quantization error.

then, we can treat E_T as a white noise with a probability density function as shown in figure 6.2.2.

From figure 6.2.2, it is observed that the maximum quantization error due to the truncation of two's complement numbers is -2^{-b} while the variance and mean of the truncation error are easily shown to be $2^{-2b}/12$ and $-2^{-b}/2$ respectively (11). Nevertheless, the choice of two's complement notation has an advantage over systems with sign-

and-magnitude notation, as sign-and-magnitude truncation (10) produces an error whose variance is four times that of two's complement truncation, that is, $2^{-2b}/3$.

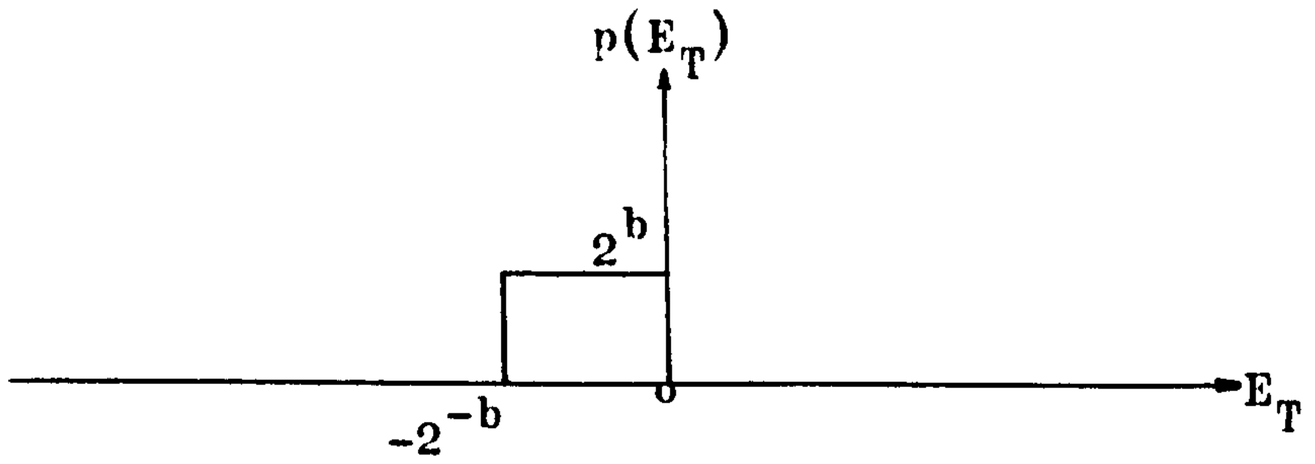


FIG: 6.2.2 PROBABILITY DENSITY FUNCTION FOR TWO'S COMPLEMENT TRUNCATION

6.2.2 EFFECT OF TWO'S COMPLEMENT ROUNDING:

As an alternative to truncation, numbers can be rounded to fit into a finite word length register. Rounding to a number of b bits is accomplished by choosing the rounded result as the b-bit number closest to the original unrounded number. After rounding, the numbers are quantized in steps of 2^{-b}. It is, of course, possible that the number to be rounded lies exactly halfway between two quantization levels. In this case, there are several possible strategies, such as always rounding up, always rounding down, or using random rounding, and a random choice ought to be made as to which strategy to use. However, in most situations, an arbitrary choice will have negligible effect on the accuracy of the resultant computation.

For two's complement arithmetic, the error made by rounding a number to (b+1) bits, including sign,

$$E_R = Q[x] - x \qquad (6.2.5)$$

satisfies the inequality:

$$-2^{-b}/2 < E_R \leq 2^{-b}/2 \qquad (6.2.6)$$

where the symbol $Q[x]$ denotes the rounded number in this case.

The above nonlinear relationship representing the rounding of two's complement numbers is depicted in figure 6.2.3.

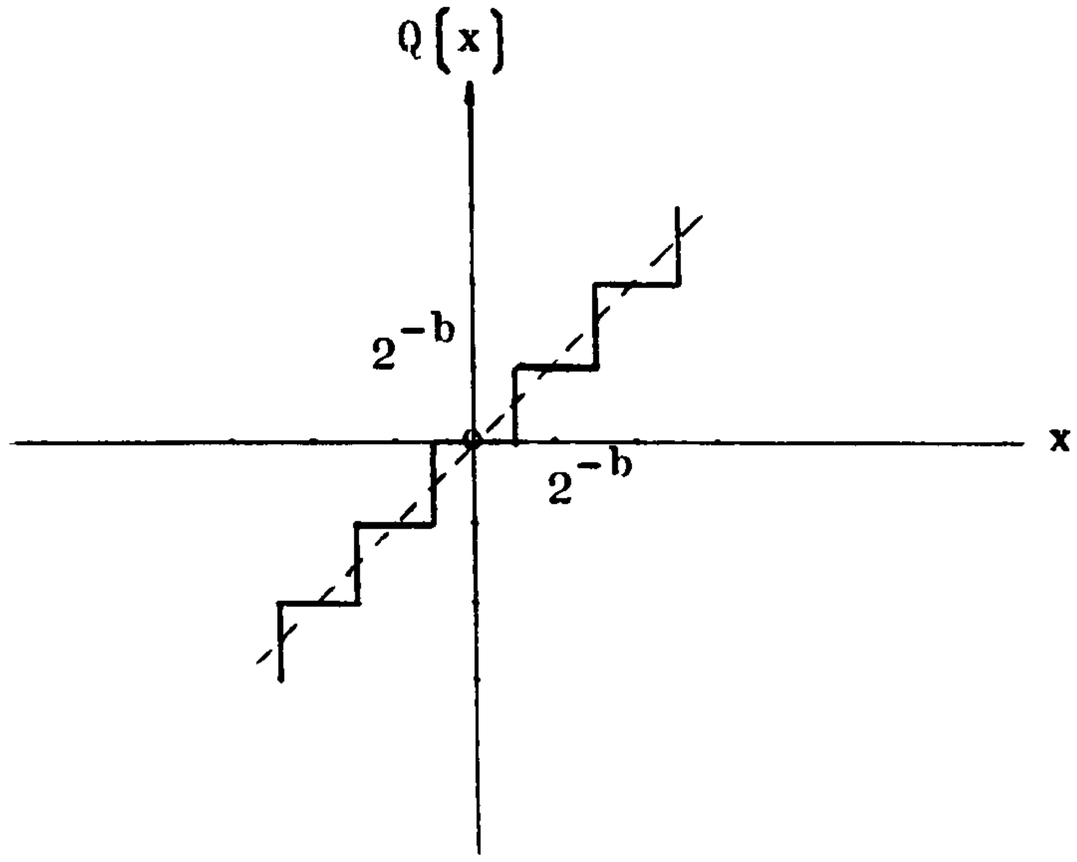


FIG: 6.2.3 NONLINEAR RELATIONSHIP REPRESENTING A TWO'S COMPLEMENT ROUNDING QUANTIZER.

Furthermore, if we make the same assumptions as in the case of truncating a two's complement number to (b+1) bits, including sign, in section 6.2.1, we can treat E_R as a white noise with a probability density function as shown in figure 6.2.4.

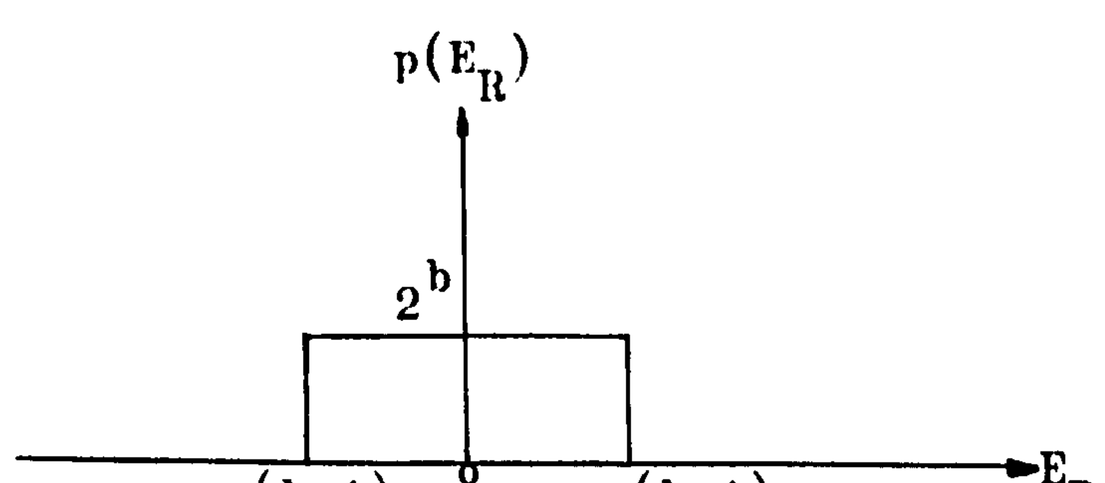


FIG: 6.2.4 PROBABILITY DENSITY FUNCTION FOR TWO'S COMPLEMENT ROUNDING

From figure 6.2.4, it is observed that the maximum rounding error is $\pm 2^{-(b+1)}$ while the variance and mean of the rounding error are easily shown to be $2^{-2b}/12$ and 0 respectively (11).

6.3 EFFECT OF INPUT QUANTIZATION ON THE PROPOSED HARDWARE IMPLEMENTATION:

In our discussion of the effect of input quantization on the proposed CBFCS approach, we have mainly followed a general approach outlined in reference (89), since this approach to the analysis of input quantization error is also applicable in our case.

Since any input signal processed by a digital filter has to be in a digital form (e.g. obtained by an analogue-to-digital conversion of an analogue input signal), the digital input signal to a digital filter is consequently subject to quantization. If the quantizer has equal step size 2^{-b}, then the input to the actual filter can be represented by

$$x'_n = x_n + \delta_n \qquad (6.3.1)$$

where δ_n is the input quantization error bounded by

$$-2^{-(b+1)} < \delta_n \leq 2^{-(b+1)} \qquad (6.3.2)$$

for two's complement rounding, and

$$0 \geq \delta_n > -2^{-b} \qquad (6.3.3)$$

for two's complement truncation.

In the following discussion, we shall assume that the above quantization error sequence $\{\delta_n\}$ is uncorrelated with the input sequence $\{x_n\}$, in addition to the previous three assumptions made in section 6.2.

Since the filter is linear, we can treat the filter output as the sum of two components: one due to the input sequence $\{x_n\}$ and the other due to the error sequence $\{\delta_n\}$ while assuming that the signal and noise are uncorrelated as mentioned. Therefore, the output noise sequence due to $\{\delta_n\}$ as an input is simply given by the convolution sum,

$$\sum_{k=0}^{n} h_k \cdot \delta_{n-k} = \sum_{k=0}^{n} h_{n-k} \cdot \delta_k \qquad (6.3.4)$$

where $\{h_n\}$ is the discrete-time impulse response of the digital filter. In fact, it is seen that the output component due to $\{\delta_n\}$ is bounded in absolute value by

$$\sum_{k=o}^{n} |h_k| . 2^{-(b+1)} \qquad (6.3.5)$$

for two's complement rounding, as the maximum rounding error is $2^{-(b+1)}$ in magnitude, and

$$\sum_{k=o}^{n} |h_k| . 2^{-b} \qquad (6.3.6)$$

for two's complement truncation, as the maximum truncation error is 2^{-b} in magnitude.

Furthermore, the steady-state output component due to $\{\delta_n\}$ is a wide-sense-stationary sequence with power spectral density (89) given by

$$H(z) . H(\frac{1}{z}) . \frac{2^{-2b}}{12} \qquad (6.3.7)$$

where $H(z)$ is the transfer function of the digital filter. In the above discussion of input quantization, we have assumed infinite precision processing within the digital filter, and, consequently, the effect on the output of coefficient inaccuracy and roundoff accumulation has been ignored.

From equation (6.3.7), the variance of the resultant error at the output due to input quantization error can be obtained by integrating the power spectral density of the noise sequence. It is given by

$$\frac{1}{2\pi j} \oint H(z) . H(\frac{1}{z}) . \frac{2^{-2b}}{12} . \frac{dz}{z} \qquad (6.3.8)$$

Alternatively, the output noise variance due to $\{\delta_n\}$ can be found by noting that the noise samples in equation (6.3.4) are statistically independent and have the same variance, viz., $2^{-2b}/12$. Thus, at any time, the output noise variance is simply the sum of the variances of individual terms in equation (6.3.4), given by

$$\frac{2^{-2b}}{12} \left[\sum_{k=o}^{n} |h_k|^2 \right] \qquad (6.3.9)$$

6.3.1 INTERRELATIONSHIP BETWEEN SIGNAL-TO-NOISE RATIO AND INPUT QUANTIZATION:

When a digital filter is used to process sampled analogue signals, the error due to input quantization is commonly viewed as an additive noise signal as mentioned previously. The ratio of the signal power to the noise power is a useful measure of the relative strengths of the signal and noise present. For a digital filter employing either of the two two's complement quantizers described in section 6.2, the available input signal-to-noise ratio is

$$SNR_{in} = \frac{V(x_n)}{V(\mathcal{J}_n)} = \frac{V(x_n)}{2^{-2b}/12} \qquad (6.3.10)$$

where the symbol V denotes the mathematical variance of a function, and $V(x_n)$ and $V(\mathcal{J}_n)$ are the values of the variances of the input sequence $\{x_n\}$ and the error sequence $\{\mathcal{J}_n\}$ respectively.

When expressed on a logarithmic scale, equation (6.3.10) becomes

$$SNR_{in} = 10.\log_{10}V(x_n)(2^{2b}.12)$$

$$= 6.02b + 10.79 + 10.\log_{10}V(x_n) \qquad (6.3.11)$$

where it is clear that the input SNR increases approximately 6dB for every additional bit used in the quantization process.

In our discussion of the effect of input quantization, we have assumed that the input signal does not exceed the dynamic range of the two's complement quantizers as defined by equation (6.2.2). If the exact value of an input sample falls outside the above dynamic range, then additional distortion due to input clipping results. This clipping of the input is undesirable and must be eliminated by scaling the amplitude of the input until the dynamic range constraint above is satisfied. Thus, we quantize scaled input samples when the input signal exceeds the dynamic range of the input quantizer, i.e., we quantize Ax_n instead of x_n where $0 < A < 1$ is a scaling constant so chosen that the value of Ax_n is normalized to that of the dynamic range of the input quantizer to maximize the input signal-to-noise ratio. Since the variance of Ax_n is $A^2.V(x_n)$, the signal-to-noise ratio of the normalized input becomes:

$$SRN_{in} = 6b + 10.8 + 10.\log_{10} V(x_n) + 20.\log_{10}(A) \qquad (6.3.12)$$

A comparison of equations (6.3.11) and (6.3.12) shows that scaling the amplitude of the input to reduce clipping distortion reduces the input signal-to-noise ratio. On the other hand, it is also feasible in theory to amplify a minute input signal to increase the input signal-to-noise ratio such that A is now a number greater than unity which satisfies the dynamic range constraint.

Similarly, the output signal-to-noise ratio can be defined by using equation (6.3.8) or (6.3.9) as below:

$$SNR_{out} = \frac{V(y_n)}{\frac{1}{2\pi j}\oint H(z).H(\frac{1}{z}).\frac{2^{-2b}}{12}.\frac{dz}{z}} \qquad (6.3.13)$$

$$= \frac{V(y_n)}{\frac{2^{-2b}}{12}\left(\sum_{k=o}^{n}|h_k|^2\right)} \qquad (6.3.14)$$

where $V(y_n)$ is the variance of the output sequence $\{y_n\}$.

In deriving these equations, we have implicitly assumed that the digital filter has been realized with infinite precision, that is, no error. In fact, this is not true in practice, due to filter coefficient quantization and roundoff accumulation which will be treated seperately in later sections of the present chapter. However, it is reasonable to assume that the errors in the output due to errors introduced in the realization of the digital filter are independent of errors due to the input quantization noise. Thus, errors due to rounding or truncation in realizing a digital filter are considered seperately and their effects superimposed on the errors due to input quantization.

So far we have discussed input quantization effect by making some statistical assumptions in the hardware implementation of the CBFCS algorithm which have their limitations. It should, however, be pointed out that it is possible to find examples where these

assumptions are clearly not valid. For instance, if the input signal is a step function, it would be impossible to justify the above assumptions. However, when the sequence $\{x_n\}$ is a complicated signal, such as speech or music, where the signal fluctuates rapidly in a somewhat unpredictable manner, these assumptions are then more realistic. Experiments have shown (86), (87), that as the signal becomes more complicated, the correlation between the signal and the quantization error decreases, and the error samples also become uncorrelated. In a heuristic sense, the assumptions appear to be valid if the signal is sufficiently complex and the quantization steps are sufficiently small, so that the amplitude of the signal is likely to traverse many quantization steps in going from sample to sample. Furthermore, if one's complement or sign-and-magnitude notation were used in the proposed CBFCS algorithm, the assumption that the error is independent of the signal is no longer valid, since the sign of the error is always opposite to the sign of the signal. However, on the other hand, the use of two's complement quantization satisfies the above assumptions.

In conclusion, our statistical approach to the analysis of the effect of input quantization has been based on a continuous model where the random error variables are implicitly assumed to be continuous. It will be, however, pointed out in a later section that this assumption will only be realistic as far as input quantization is concerned, and when considering the roundoff errors due to multiplications, a discrete model would be more appropriate where the random error samples are discrete variables.

6.4 COEFFICIENT QUANTIZATION IN THE PROPOSED HARDWARE IMPLEMENTATION:

In general, the coefficients of a digital filter are obtained by some theoretical design procedure that essentially assumes infinite precision representation of the filter parameters. Likewise, the proposed design technique for realtime digital filtering, as previously mentioned in chapter three, also assumes an infinite precision representation of the filter coefficients. However, in hardware implementations, the coefficients of the filters must be quantized to a finite number of bits, thereby losing accuracy. Thus,

the frequency response of the actual digital filter implemented deviates from the ideal or desired frequency response where all filter coefficients are represented with infinite precision.

Briefly, the general approaches to the analysis and synthesis of digital filters with finite word length coefficients can be divided into two categories. The first approach is to treat the coefficient quantization errors as statistical quantities whereby the resultant effects of coefficient inaccuracy due to quantization can be represented by a "stray transfer function" in parallel with the corresponding ideal filter (92), (93). Also, by making certain statistical assumptions about the coefficient quantization errors, the expected mean-squared difference (error variance) between the frequency responses of the actual and ideal filters can be readily calculated. On the other hand, the second approach to coefficient quantization is to look into individual designs to optimize their finite precision coefficients, in order to minimize the maximum weighted difference between the ideal and actual frequency responses (94)-(101). Although, in actual fact, one cannot make any general statement about coefficient quantization, this optimization approach has the advantage of yielding the best finite precision representation of the ideal or desired frequency response.

Furthermore, results from previous work, as given in the above references, have demonstrated that the parallel form of realization of high order digital filters as a combination of basic first- or second-order digital filters yields the best performance in terms of finite register lengths required in hardware implementations, or when the digital filters are realized on a general-purpose computer. On the other hand, coefficient inaccuracy has profound effects in high order digital filters implemented in the direct form. As a result, the proposed design technique described in chapter three also employs the parallel form of realization in that a desired filter is being made up of a number of elemental filters (second-order resonators) in parallel.

Prior to proposing a quantized design procedure, we shall next consider the necessary accuracy for the finite word length representations of the coefficients of the elemental filters in the proposed design technique for realtime digital filters in chapter 3.

6.4.1 ERRORS DUE TO COEFFICIENT QUANTIZATION IN THE PROPOSED DESIGN TECHNIQUE FOR REALTIME DIGITAL FILTERS:

Consider a second-order elemental filter with resonance frequency ω_r:

$$\frac{1 - R(\cos\omega_r T)z^{-1}}{1 - 2R(\cos\omega_r T)z^{-1} + R^2 . z^{-2}} \qquad (6.4.1)$$

The coefficient quantization problem of such a second-order filter has received extensive treatment in references (44), (102), (103) and we shall relate their results to our proposed design technique for realtime digital filters.

From equation (6.4.1), the angular position of the pole-pair will be $\theta = \omega_r T$. Parametric errors will cause an error in the product $\omega_r T$. Thus, if, for instance, the sampling frequency is doubled or T being halved, the resonant frequency error will tend to double for comparable errors in the coefficients of the z^{-1} and z^{-2} terms in the denominator of equation (6.4.1). To realize a given filter, it then appears that the use of high sampling rates will require greater accuracy of computation, since an increase in sampling frequency simply implies moving the poles towards the point $z = 1$. Thus, the above coefficient quantization error will finally set a limit of the sampling rate in a particular design, as T is descreased beyond a certain point, the digital filter response will deviate from the expected value.

Moreover, explicit expressions for the errors in the pole positions caused by quantization of the filter coefficients can be derived for equation (6.4.1) if it is assumed that the errors are small, so that,

$$\Delta R = \frac{\partial R}{\partial b_2}.\Delta b_2 + \frac{\partial R}{\partial b_1}.\Delta b_1 \qquad (6.4.2)$$

$$\Delta \theta = \frac{\partial \theta}{\partial b_2}.\Delta b_2 + \frac{\partial \theta}{\partial b_1}.\Delta b_1 \qquad (6.4.3)$$

giving,

$$\Delta R = \frac{1}{2R}.\Delta b_2 \qquad (6.4.4)$$

$$\Delta\theta = \frac{\Delta b_2}{2R^2.\tan\theta} - \frac{\Delta b_1}{2R.\sin\theta} \qquad (6.4.5)$$

where $b_1 = 2R.\cos\omega_r T$, and $b_2 = R^2$.

Since $\theta = T.\Delta\omega_r$, equation (6.4.5) shows directly that the error sensitivity is directly proportional to the sampling rate. Furthermore, it is clear that errors in the angle θ, and hence the resonant frequency ω_r, are greater when θ is small such that, when T and θ approach zero, equation (6.4.5) becomes

$$\Delta\theta = \frac{\Delta b_2}{2R^2\theta} - \frac{\Delta b_1}{2R\theta} \qquad (6.4.6)$$

From the above equations, the changes in pole positions and their sensitivity to coefficient quantization can be worked out. However, in the next section, we propose a quantized design procedure which is a modification of the optimization procedure previously mentioned in section 3.2.6 in chapter three.

6.4.2 A QUANTIZED DESIGN PROCEDURE:

The proposed quantized design procedure is a modification of the optimization procedure in section 3.2.6, to accommodate the effects of coefficient quantization. Consider the optimization model in figure 6.4.1 for our quantized procedure where we define

$$E(\omega) = \frac{\left|\left|H_D(e^{j\omega T})\right| - \left|H_Q(e^{j\omega T})\right|\right|}{\left|\Delta H(e^{j\omega T})\right|} \leqslant 1 \qquad \forall\omega \qquad (6.4.7)$$

as an error criterion, and $E(\omega)$ is the error index.

For every satisfactory design, the error set

$$\left|\left|H_D(e^{j\omega T})\right| - \left|H_Q(e^{j\omega T})\right|\right| \qquad (6.4.8)$$

should satisfies the tolerance set $\Delta H(e^{j\omega T})$, defined as shown in

figure 6.4.1, such that the inequality (6.4.7) holds, and $H_Q(e^{j\omega T})$ is the resultant quantized design of the desired transfer function $H_D(e^{j\omega T})$.

In chapter three, we had used the optimization procedure in section 3.2.6 to search for an optimum transfer function in a continuous parametric space where the optimized transfer function had assumed infinite precision representation in its coefficients. In contrast, the proposed quantized design procedure in this chapter basically makes use of the above optimization procedure in a discrete parametric space to determine the finite precision coefficients of the quantized filter transfer function $H_Q(e^{j\omega T})$ that satisfies the inequality (6.4.7) best, that is, when the error index $E(\omega) \simeq 1$. The quantized design procedure is outlined below:

(1) Design the desired filter to specifications, using infinite precision for coefficients, thus obtaining $H_D(e^{j\omega T})$. From the given specifications, define a tolerance set

$$\Delta H(e^{j\omega T}) = \{\Delta H_P(e^{j\omega T}), \Delta H_S(e^{j\omega T}), \Delta H_T(e^{j\omega T})\} \tag{6.4.9}$$

for the allowable errors in the passband, stopband and transition band respectively as shown below:

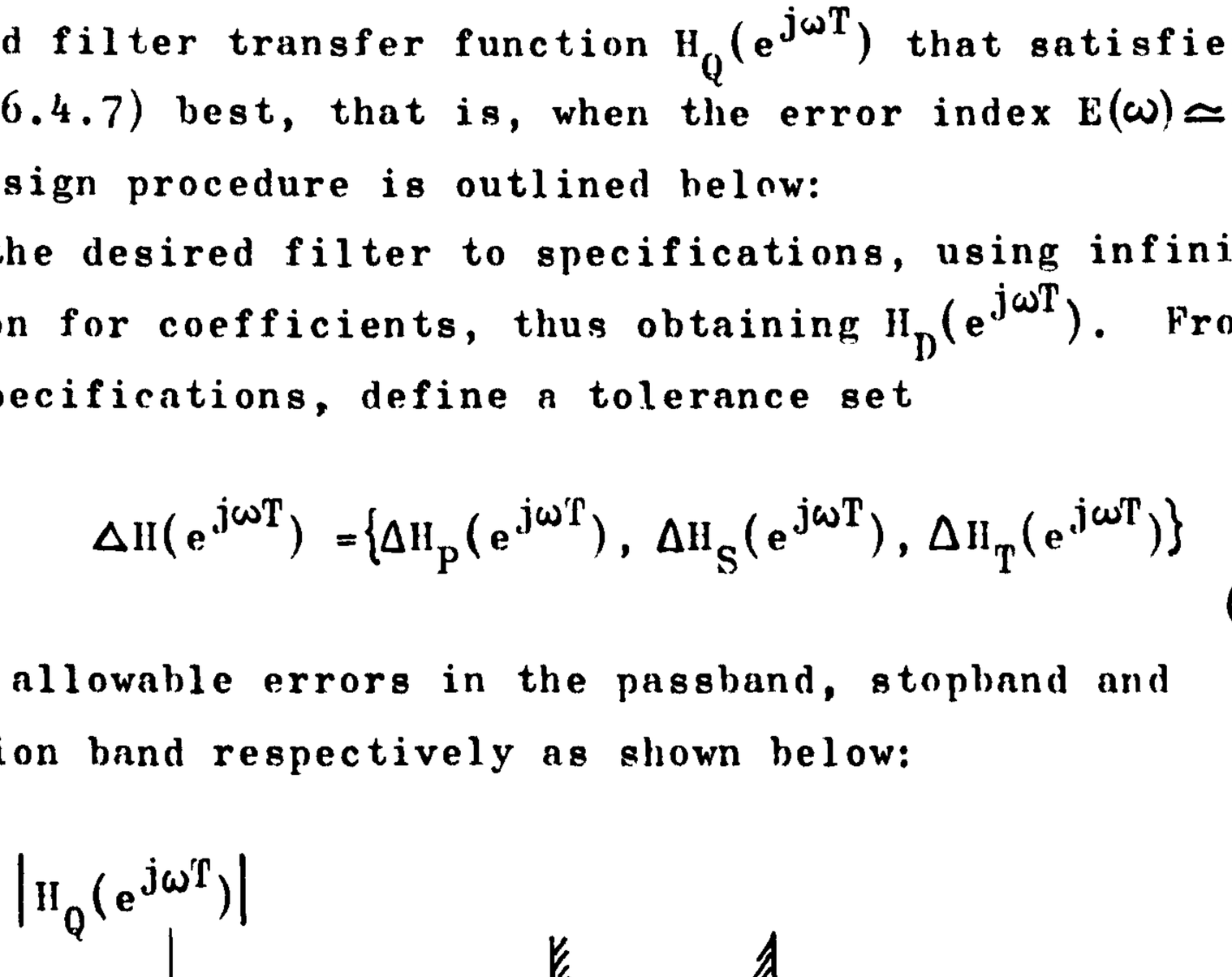


FIG: 6.4.1 OPTIMIZATION MODEL FOR A QUANTIZED DESIGN

(2) Quantized the set of infinite precision coefficients obtained in step one. Re-compute the above frequency response using the optimization procedure in section 3.2.6, and the above quantized coefficients so as to arrive at an initial quantized response. The initial word length used above is chosen by first rounding the infinite precision coefficients in step one to an arbitrarily long word length. Now, two possibilities arise. In the first instance, if the initial quantized response fails to meet the tolerance set given by equation (6.4.9), even after optimization, then we increase the present word length by one bit. However, if in the second case, the initial quantized response meets the tolerance set after optimization, then we decrease the present word length by one bit instead. In both cases above, however, we repeat the process until the computed quantized response just meets the specifications laid down. At this stage, the error index $E(\omega)$ should be approximately equal to unity implying that no further improvements can be made with the present set of quantized coefficients thus obtained.

(3) Using the finite word length obtained in step two above, compute all possible quantized pole locations and construct a discrete space or grid structure in the z-plane where we now furtherapply the optimization procedure in section 3.2.6 to search for a final quantized design.

(4) With the quantized coefficients obtained in step two above, compute the corresponding set of quantized pole locations and locate them in the discrete space or grid structure constructed in step three. Each of these poles is now moved in turn onto a new quantized location bounded by a circle centred at its present location, with a radius defined by the present quantization step. The sets of new quantized coefficients which result from the total possible combinations of the new pole locations, are now used in turn in the above discrete space optimization procedure to search for the set of new quantized coefficients that gives the best quantized response. If the error index in this case is not substantially less than unity, then retain this set of quantized coefficients as the final quantized design coefficients. If, on the other hand, the error index is substantially less than

unity, decrease the present word length by one bit and repeat the above steps until a final quantized design is reached.

Figure 6.4.2 below shows an example of a 3-bit quantization grid structure for the proposed design technique for realtime digital filters in chapter three.

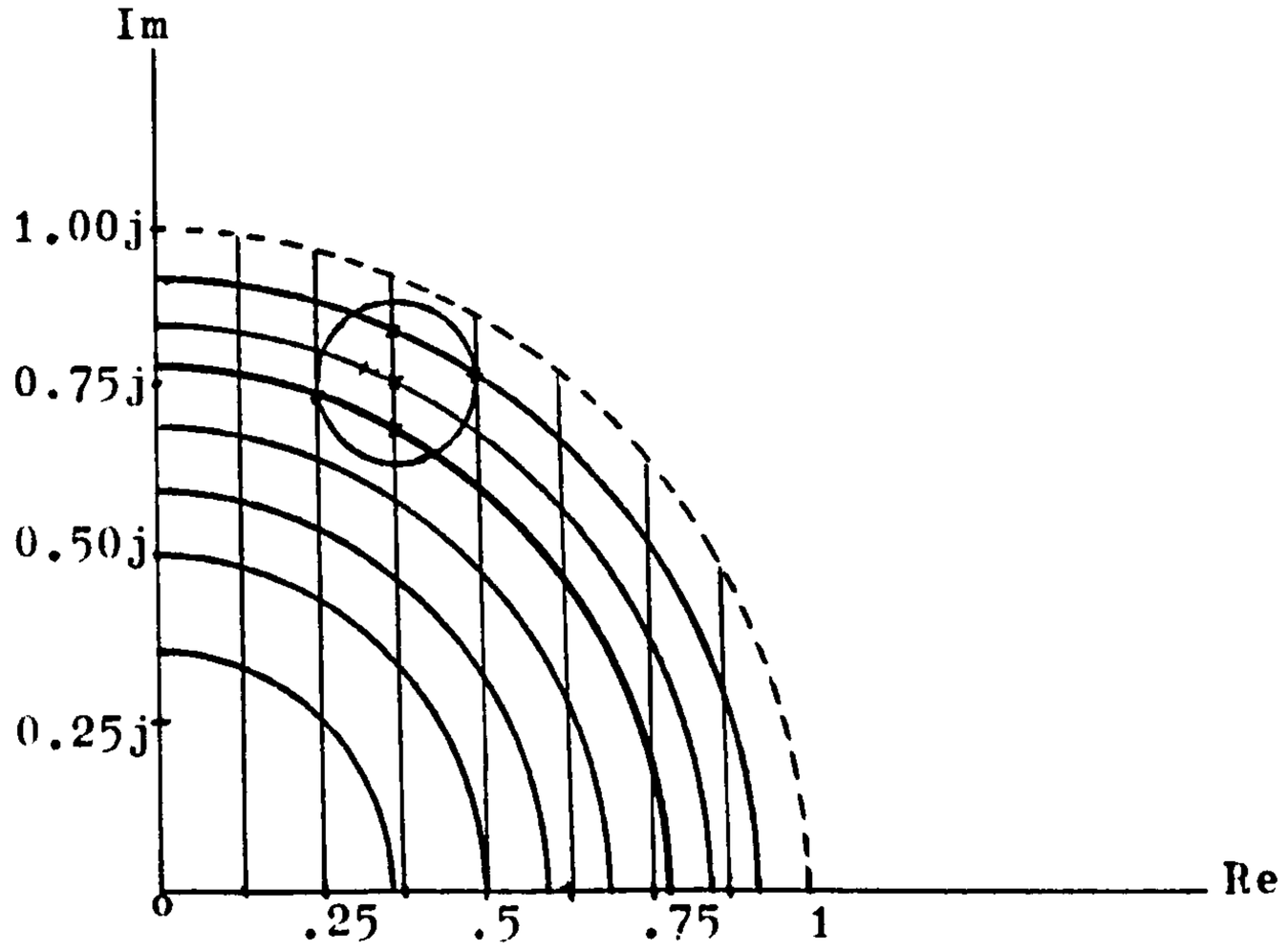


FIG: 6.4.2 A Z-PLANE DISCRETE SPACE FORMED BY A GRID OF ALL POSSIBLE QUANTIZED (3-BIT) POLE LOCATIONS OF AN ELEMENTAL FILTER

Suppose a second-order elemental filter of the proposed design technique discussed in chapter three has one of its pole at the centre of the circle shown in figure 6.4.2. It is seen that this pole has four adjacent poles bounded by the circle whose radius has the value of a quantization step. These four adjacent quantized poles are used in turn in step four above. The two adjacent poles within the circle are used first followed by the two poles on the circle.

The above grid structure is formed in the z-plane by simply intersecting concentric circles of radii R, where

$$R = e^{-aT} \qquad (6.4.10)$$

and lines parallel to the imaginary axis, at a distance of $R\cos\theta$

where, for the elemental filters in the proposed design technique in chapter three,

$$\cos\theta = \begin{cases} \cos(\omega_{-p} - nb)T & \text{in the lower transition band.} \\ \cos(\omega_{-p} + 2kb)T & \text{in the passband.} \\ \cos(\omega_{+p} + nb)T & \text{in the upper transition band.} \end{cases} \tag{6.4.11}$$

A feature of the above quantized pole-grid for fixed-point representation, is the non-uniform distribution of the quantized pole locations and their concentration near the unit circle around the imaginary axis. It has the significance that more possible quantized pole locations are available for narrow-band bandpass elemental filters for the proposed design technique discussed in chapter three. However, the underlying sampling frequency should be so chosen that θ in the above equation will not be small. This follows from our discussion in section 6.4.1.

In addition to the above mentioned, compromises should always be made between the use of higher order filters to achieve design specifications and the minimum order required, if the coefficient quantization requirement for a lower order filter proves too stringent to meet. In fact, there is a tradeoff between coefficient word length and the order of the resultant filter. This is valid since the error index $E(\omega)$ must be less than unity for the infinite precision filter (or else no finite precision filter will be acceptable), and one can always reduce the value of $E(\omega)$ by increasing the filter order. Thus, the question arises as to whether a higher order filter can be quantized with a smaller word length than a lower order filter when the initial value of $E(\omega)$ is small when compared to unity in step one of the proposed quantized design.

To summarize, we have thus proposed a quantized design procedure above which makes use of an optimization method in a discrete space to search for an optimum quantized design. The resultant quantized filter, when implemented in hardware via the proposed CBFCS algorithm, will become asymptotically linear as the word length for the filter variables increases. So far, in the

previous sections, we have dealt with the problems of input quantization and coefficient quantization, including a discussion on the coefficient sensitivity of the elemental filters in the proposed design technique in chapter three. In the next section, we shall describe a discrete roundoff noise model used in the analysis of product quantization or multiplication roundoff in the proposed hardware implementation.

6.5 A DISCRETE ROUNDOFF NOISE MODEL FOR PRODUCT QUANTIZATION IN THE PROPOSED HARDWARE IMPLEMENTATION:

As previously pointed out in section 6.3, the continuous noise model, in which the error samples are treated as continuous random variables, is only realistic in the analysis of the effect of input quantization where the usual approach is to assume infinite precision arithmetic within the digital filter. However, in the analysis of product quantization or multiplication roundoff, the error associated with a quantized product is a discrete variable so that the usual continuous noise model is now non-realistic. Hence, we propose the use of a discrete roundoff noise model in the analysis of product quantization in the proposed hardware implementation, such that the error samples are now treated as discrete random variables

Owing to product quantization, each output sample y_n in the proposed CBFCS implementation can be regarded as a quantized product obtained by the summation of a finite number of partial products. Here, we assume that the transfer function $H(z)$ implemented by the proposed CBFCS approach is designed with the quantized procedure in section 6.4.2 such that each partial product above is exact (i.e. no coefficient quantization error). Provided that no overflows occur, the above summation will not lead to any error. However, owing to a finite word length , the final product y_n has to be quantized, thus yielding a product quantization error. In our discrete roundoff noise model, y_n is assumed to be rounded. Further assumptions made are:

(1) The error samples γ_n in the error sequence $\{\gamma_n\}$ are statistically independent random variables.

(2) For each n, γ_n is a discrete random variable with possible values in an error set $\{E_k\}$ such that each member E_k has the same

probability of occurence, that is, a uniform, flat probability density distribution.

Consider the general case of the hardware implementation of the CBFCS algorithm where the filter coefficients and variables are bounded by ± 1 and represented in two's complement notation. Let (m+1) bits, including sign, be used to represent the variables and (n+1) bits, including sign, for the coefficients. The output of the filter

$$y_n = -y_n^0 + \sum_{j=1}^{m+n} y_n^j . 2^{-j} \qquad (6.5.1)$$

is up-rounded, as mentioned above, to become a (b+1)-bit number

$$y'_n = -y'^0_n + \sum_{j=1}^{b} y'^j_n . 2^{-j} \qquad (6.5.2)$$

by truncating

$$y_n + 2^{-(b+1)}$$

to (b+1) bits where $b < m+n$.

Let γ_n be a discrete error variable where

$$y'_n = y_n + \gamma_n \qquad (6.5.3)$$

is a member of the output error sequence $\{\gamma_n\}$ whose values are represented in the error set $\{E_k\}$. Now, if we further lay down the restriction that in up-rounding the above product y_n, all "end effects" due to the asymmetry in two's complement representations are excluded, then the discrete error variable γ_n can only have values in the error set

$$\{E_k\} = \left\{k.2^{-(m+n)} : k = 0, \pm 1, \ldots, -2^{m+n-(b+1)}\right\} \qquad (6.5.4)$$

"end effects" will be discussed later.

The above error set $\{E_k\}$ in equation (6.5.4) can be arrived at by considering the up-rounding process in two seperate cases:

(1) if the (b+1)th bit of y_n is zero, then the error due to up-rounding, γ_n, is positive and has $2^{m+n-b-1}$ different values:

$$0 \leqslant \gamma_n \leqslant 2^{-(m+n)}(2^{m+n-(b+1)} - 1) \qquad (6.5.5)$$

(2) if the (b+1)th bit of y_n is one, then the error due to up-rounding, γ_n, is negative and has $2^{m+n-b-1}$ different values:

$$-2^{-(m+n)} \cdot 2^{m+n-(b+1)} \leqslant \gamma_n \leqslant -2^{-(m+n)} \qquad (6.5.6)$$

In the set $\{E_k\}$, the element for which $k=-2^{m+n-(b+1)}$ has the value of $-2^{-(b+1)}$. Furthermore, $\{E_k\}$ contains the element $-2^{-(b+1)}$ but not the element $2^{-(b+1)}$ due to the fact that any y_n that lies $2^{-(b+1)}$ units away from any quantization level of y'_n will be assigned to the next higher quantization level in the up-rounding process. On the other hand, any y_n that lies $2^{-(b+1)}$ units away from any quantization level of y'_n will be assigned to the next lower level in a down-rounding process.

From equation (6.5.4), it is seen that $\{E_k\}$ contains 2^{m+n-b} members, or an even number of members. By assumption (2) above, the error samples in $\{E_k\}$ are equi-probable, and the probability of any one sample in $\{E_k\}$ is thus $2^{-(m+n-b)}$. For up-rounded two's complement representation, the element $2^{-(b+1)}$ is generated only in one instance as an "end effect". When signal dynamic range is properly controlled, the occurence of this end effect will be relatively infrequent. Nevertheless, assumption (2) will become inappropriate if the element $2^{-(b+1)}$ were included in the error set $\{E_k\}$ above.

Table 6.5.1 illustrates the above "end effects" and shows all possible values of the sequences $\{y_n\}$, $\{y'_n\}$ and $\{\gamma_n\}$ for m=n=b=2. From table 6.5.1, neglecting "end effects", it is clear that $\{E_k\}$ contains the value $-2^{(b+1)}=-0.125$ but not the value $2^{-(b+1)}=0.125$. It is further noticed that the value 0.125 is generated only once among the 32 possible values of $\{y_n\}$. By neglecting the "end effects", the four possible error samples in the error set $\{0.0625, 0, -0.0625, -0.125\}$ occur an equal number of times, and may thus be regarded as equi-probable. This is, of course, approximately true, unless all values of y_n with these error values are equally likely.

FINAL PRODUCT y_n		UP-ROUNDED VALUE y'_n		ROUNDOFF ERROR γ_n		
0.9375	0.1111	0.75	0.11	0.1875	0.0011	UPPER END EFFECTS
0.875	0.1110	0.75	0.11	0.125	0.0010	
0.8125	0.1101	0.75	0.11	0.0625	0.0001	
0.75	0.1100	0.75	0.11	0	0.0000	
0.6875	0.1011	0.75	0.11	-0.0625	1.1111	
0.625	0.1010	0.75	0.11	-0.125	1.1110	
0.5625	0.1001	0.5	0.10	0.0625	0.0001	
0.5	0.1000	0.5	0.10	0	0.0000	
0.4375	0.0111	0.5	0.10	-0.0625	1.1111	
0.375	0.0110	0.5	0.10	-0.125	1.1110	
0.3125	0.0101	0.25	0.01	0.0625	0.0001	
0.25	0.0100	0.25	0.01	0	0.0000	
0.1875	0.0011	0.25	0.01	-0.0625	1.1111	
0.125	0.0010	0.25	0.01	-0.125	1.1110	
0.0625	0.0001	0	0.00	0.0625	0.0001	
0	0.0000	0	0.00	0	0.0000	
-0.0625	1.1111	0	0.00	-0.0625	1.1111	
-0.125	1.1110	0	0.00	-0.125	1.1110	
-0.1875	1.1101	-0.25	1.11	0.0625	0.0001	
-0.25	1.1100	-0.25	1.11	0	0.0000	
-0.3125	1.1011	-0.25	1.11	-0.0625	1.1111	
-0.375	1.1010	-0.25	1.11	-0.125	1.1110	
-0.4375	1.1001	-0.5	1.10	0.0625	0.0001	
-0.5	1.1000	-0.5	1.10	0	0.0000	
-0.5625	1.0111	-0.5	1.10	-0.0625	1.1111	
-0.625	1.0110	-0.5	1.10	-0.125	1.1110	
-0.6875	1.0101	-0.75	1.01	0.0625	0.0001	
-0.75	1.0100	-0.75	1.01	0	0.0000	
-0.8125	1.0011	-0.75	1.01	-0.0625	1.1111	
-0.875	1.0010	-0.75	1.01	-0.125	1.1110	
-0.9375	1.0001	-1	1.00	0.0625	0.0001	LOWER END EFFECTS
-1	1.0000	-1	1.00	0	0.0000	

TABLE 6.5.1.

Note that the "end effects", if not excluded, will spoil the above symmetry shown in table 6.5.1. The "upper end effects" are the errors 0.1875 and 0.125 since up-rounding the two's complement numbers 0.1110 (0.875) and 0.1111 (0.9375) will yield 1.00 which cannot be represented in the above limited dynamic range. Hence, in the actual CBFCS hardware implementation, saturation logic is required in addition to rounding logic to prevent the upper values of the sequence $\{y_n\}$ from up-rounding to negative numbers which will further induce overflow errors (saturation arithmetic and overflow errors will be discussed in a later section). In neglecting the "upper end effects", we also neglect the "lower end effects" in our model in order to assume that the four error samples in the error set $\{E_k\}$ are equi-probable. It is appropriate to neglect the "end effects" if large output sample values of y_n are much less likely than smaller values (for example, by properly controlling the dynamic range of the output signal). This is typically the case when the word length of the up-rounded two's complement number y'_n is at least several bits.

Mathematically, the variance of the roundoff error γ_n is given by

$$V(\gamma_n) = \mathcal{E}((\gamma_n - \mathcal{E}(\gamma_n))^2)$$

$$= \mathcal{E}(\gamma_n^2) - (\mathcal{E}(\gamma_n))^2 \qquad (6.5.7)$$

where $\mathcal{E}$ is the statistical expectation. From equations (6.5.4), (6.5.5) and (6.5.6) above, the mean value of the discrete error variable γ_n can be found as

$$\mathcal{E}(\gamma_n) = 2^{-(m+n-b)} \sum_{k=-2^{m+n-(b+1)}}^{2^{m+n-(b+1)}-1} k.2^{-(m+n)}$$

$$= -2^{-(m+n+1)} \qquad (6.5.8)$$

and

$$\mathcal{E}(\gamma_n^2) = \sum_{k=-\alpha}^{\alpha-1} \frac{E_k^2}{2^{m+n-b}} \qquad (6.5.9)$$

where $\alpha = 2^{m+n-(b+1)}$ and $E_k = k.2^{-(m+n)}$.

Equation (6.5.9) can also be written as two seperate sums:

$$\mathcal{E}(\gamma_n^2) = \sum_{k=1}^{\infty} \frac{((k-1)2^{-(m+n)})^2}{2\alpha} + \sum_{k=1}^{\infty} \frac{(-k.2^{-(m+n)})^2}{2\alpha} \qquad (6.5.10)$$

where the first sum is associzted with the positive elements and the zero element in the error set $\{E_k\}$ while the second sum relates to the negative elements. After further manipulations, equation (6.5.10) becomes

$$\mathcal{E}(\gamma_n^2) = \frac{2^{-2(m+n)}}{2\alpha} \sum_{k=1}^{\infty} 2k^2 - 2k + 1$$

$$= \frac{2^{-2b}}{12}(1 + 2^{2(b-m-n)+1}) \qquad (6.5.11)$$

Substituting equations (6.5.8) and (6.5.11) in (6.5.7), we have,

$$V(\gamma_n) = \frac{2^{-2b}}{12}(1 + 2^{2(b-m-n)+1}) - 2^{-2(m+n+1)}$$

$$= \frac{2^{-2b}}{12}(1 - 2^{-2(m+n-b)}) \qquad (6.5.12)$$

Next, we compare the performance of the above discrete roundoff noise model with the usual continuous model by calculating the ratio:

$$\frac{\text{DISCRETE ROUNDOFF NOISE VARIANCE}}{\text{CONTINUOUS ROUNDOFF NOISE VARIANCE}} = \frac{2^{-2b}/12(1-2^{-2(m+n-b)})}{2^{-2b}/12}$$

$$= 1 - 2^{-2(m+n-b)} \qquad (6.5.13)$$

Since the ratio in equation (6.5.13) is less than unity, the roundoff noise variance obtained with the usual continuous model can, therefore, be regarded as an upper bound on the discrete roundoff noise variance obtained in the present model using up-rounding two's complement arithmetic. Furthermore, it is concluded the error produced by multiplication roundoff is less important than other quantization noise sources. In addition, when m=b, as in the case of the proposed CBFCS hardware implementation, the ratio in equation (6.5.13) becomes $(1 - 2^{-2n})$ which purely depends on the word length of the filter coefficients.

6.6 A GENERALIZED QUANTIZATION NOISE MODEL:

In the previous sections, we have examined the various aspects of finite word length effects on the proposed CBFCS hardware implementation when the quantization errors are uncorrelated. Also, appropriate findings have been given, and as a result of these findings, we have arrived at a generalized quantization noise model for the error analysis of the proposed CBFCS hardware implementation. Since high-order digital filters are always preferred to be realized as a parallel combination of, or a cascade of first- and second-order filters, we shall base our model on a second-order difference equation.

In section 5.2, we discussed the implementation of a general second-order difference equation where we implicitly assumed infinite precision representations of the filter variables and coefficients. However, in the implementation of equation (5.2.14), y_n, which requires infinite word length, is not computed due to the effects of finite word length. As a result, an approximation y'_n is obtained by either rounding or truncating the right-hand-side of equation (5.2.14) to M bits. The error γ_n introduced by the quantizer is defined by:

$$y'_n = \gamma_n + \left[-f\left(x^o_n,\ x^o_{n-1},\ x^o_{n-2},\ y'^o_{n-1},\ y'^o_{n-2},\ C^1_{B,n},\ C^1_{2B,n},\ldots,\ C^1_{\left[\frac{N}{B}-1\right]B,n}\right) + \sum_{j=1}^{M-1} 2^{-j}.f\left(x^j_n,\ x^j_{n-1},\ x^j_{n-2},\ y'^j_{n-1},\ y'^j_{n-2},\ C^{j+1}_{B,n},\ldots,\ C^{j+1}_{\left[\frac{N}{B}-1\right]B,n}\right) + f'_n + f''_n \right] \qquad (6.6.1)$$

where the two's complement number in the above square bracket is being quantized to M bits and, γ_n satisfies the following inequalities:

$$0 \geqslant \gamma_n > -2^{-(M-1)} \qquad \text{for two's complement truncation,}$$

$$-2^{-M} < \gamma_n \leqslant 2^{-M} \qquad \text{for two's complement rounding.}$$

In equation (6.6.1), we have assumed the input x_n is exactly represented. However, due to analogue-to-digital conversion or other inexact representations of x_n, the actual input to the digital filter has become x'_n instead, where

$$x'_n = x_n + \delta_n \qquad (6.6.2)$$

is the two's complement quantized representation of x_n to M bits and δ_n is the input quantization error which satisfies the following relationships:

$$0 \geqslant \delta_n > -2^{-(M-1)} \qquad \text{for two's complement truncation,}$$

$$-2^{-M} < \delta_n \leqslant 2^{-M} \qquad \text{for two's complement rounding.}$$

Figure 6.6.1 depicts a block diagram used to study the effects of quantization errors in a general second-order digital filter implemented by the proposed CBFCS algorithm.

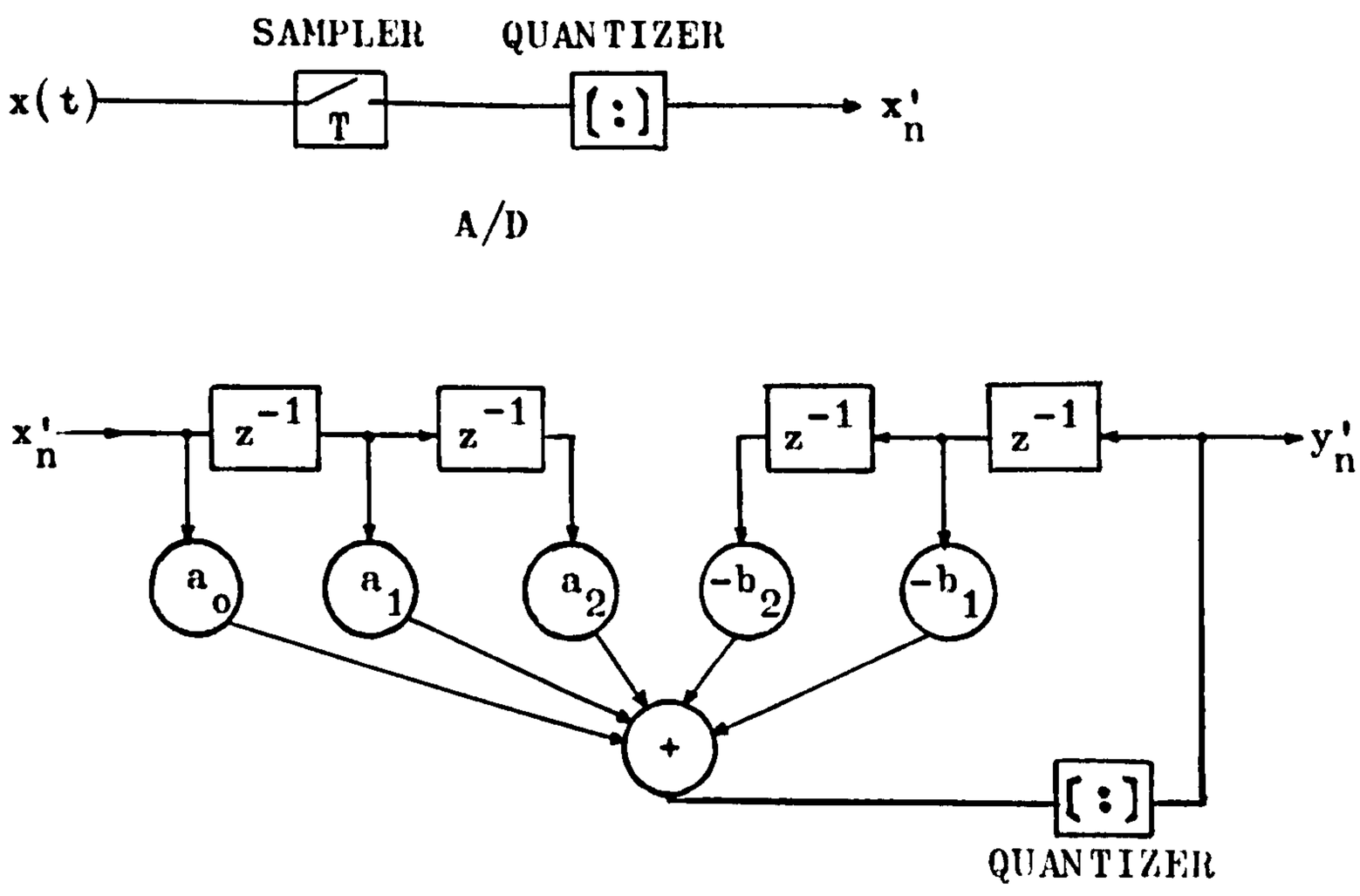


FIG: 6.6.1 BLOCK DIAGRAM OF A GENERAL SECOND-ORDER QUANTIZED DIGITAL FILTER IMPLEMENTED BY THE PROPOSED CBFCS ALGORITHM.

In the formulation of equation (6.6.1), we have implicitly assumed that the basic second-order digital filter has been designed by the proposed quantized design procedure discussed in section 6.4.2 such that the filter coefficients shown in figure 6.6.1 are exactly represented by N-bit two's complement numbers. Furthermore, in figure 6.6.1, we have also implicitly assumed that the error at the filter output due to multiplication roundoff is statistically independent of the input quantization error so that the quantizers shown are uncorrelated and can therefore be treated seperately. The total output quantization error is then the combined effects of the above two quantizers. Nevertheless, when considering the output quantizer in figure 6.6.1, the input quantizer is ignored such that the input to the second-order section is then x_n (instead of x'_n as shown) which is assumed to be exact. Similarly, when considering the input quantizer, the output quantizer is ignored and the input to the second-order section is x'_n (which is inexact) as shown. As a consequence of the above assumptions, the effects of quantization on the basic second-order section, implemented by the proposed CBFCS approach, can now be modeled with only one output quantizer due to the multiplication roundoff error γ_n and an input quantizer due to the input quantization error δ_n.

In figure 6.6.1, it is shown that the basic second-order digital filter, implemented by the proposed CBFCS approach, is always in the direct form. Let e_n be the total accumulation of quantization errors defined by the relationship:

$$y'_n = y_n + e_n \tag{6.6.3}$$

Neglecting input quantization error, e_n can be shown to satisfy the following equation:

$$e_n = \gamma_n - \sum_{i=1}^{K=2} b_i . e_{n-i} \tag{6.6.4}$$

since the error at the output of the second-order section is seen to pass through the poles

$$P(z) = 1 + \sum_{i=1}^{K=2} b_i . z^{-i} \tag{6.6.5}$$

of the transfer function

$$H(z) = \frac{\sum_{i=0}^{K=2} a_i . z^{-i}}{1 + \sum_{i=1}^{K=2} b_i . z^{-i}} \qquad (6.6.6)$$

of the quantized filter only.

By the results obtained in the previous sections, we have, then mean value of the error e_n at the output of the second-order quantized filter as

$$\mathcal{E}(e_n) = \begin{cases} -\dfrac{2^{-M}}{1 + \sum_{i=1}^{K=2} b_i} & \text{for two's complement trunction,} \\ 0 & \text{for rounding (continuous model),} \\ -\dfrac{2^{-(M+N-1)}}{1 + \sum_{i=1}^{K=2} b_i} & \text{for rounding (discrete model).} \end{cases} \qquad (6.6.7)$$

where $\mathcal{E}$ is the mathematical expectation.

The mean-square error at the output is given by

$$\mathcal{E}(e_n^2) = (\mathcal{E}(e_n))^2 + \frac{2^{-2M}}{3} \cdot \frac{1}{2\pi j} \oint_{|z|=1} \frac{1}{\left| 1 + \sum_{i=1}^{K=2} b_i . z^{-i} \right|^2} \cdot \frac{dz}{z} \qquad (6.6.8)$$

$$= (\mathcal{E}(e_n))^2 + \frac{2^{-2M}}{3} \cdot \frac{1}{2\pi j} \oint_{|z|=1} \frac{1}{P(z).P(z^{-1})} \cdot \frac{dz}{z} \qquad (6.6.9)$$

for two's complement truncation and rounding (continuous model), while for two's complement rounding (discrete model), we have

$$\mathcal{E}(e_n^2) = (\mathcal{E}(e_n))^2 + \frac{2^{-2M}}{3}(1-2^{-2(N-1)}) \frac{1}{2\pi j} \oint_{|z|=1} \frac{1}{\left| 1 + \sum_{i=1}^{K=2} b_i . z^{-i} \right|^2} \cdot \frac{dz}{z} \qquad (6.6.10)$$

$$= (\mathcal{E}(e_n))^2 + \frac{2^{-2M}}{3}(1-2^{-2(N-1)}) \frac{1}{2\pi j} \oint_{|z|=1} \frac{1}{P(z).P(z^{-1})} \cdot \frac{dz}{z} \qquad (6.6.11)$$

Therefore, the variance of the output error e_n, neglecting input error, is given by

$$V(e_n) = \mathcal{E}((e_n - \mathcal{E}(e_n))^2)$$

$$= \mathcal{E}(e_n^2) - (\mathcal{E}(e_n))^2 \qquad (6.6.12)$$

where $\mathcal{E}(e_n^2)$ and $\mathcal{E}(e_n)$ are given by equations (6.6.7)-(6.6.11) above for the quantization processes: two's complement truncation, two's complement rounding (continuous model) and two's complement rounding (discrete model).

So far, we have neglected the error due to input quantization whereby e_n includes the additional error variance

$$\frac{2^{-2M}}{3} \cdot \frac{1}{2\pi j} \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z} \qquad (6.6.13)$$

since the error J_n due to input quantization, with variance $\frac{2^{-2M}}{3}$, passes through the entire second-order section.

Therefore, the total accumulation of quantization errors, e_n, including the effect of input quantization, has a variance

$$V(e_n) = \frac{2^{-2M}}{3} \cdot \frac{1}{2\pi j} \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z}$$

$$+ \frac{2^{-2M}}{3} \cdot \frac{1}{2\pi j} \oint_{|z|=1} \frac{1}{P(z).P(z^{-1})} .\frac{dz}{z} \qquad (6.6.14)$$

for two's complement truncation and rounding (continuous model), and

$$V(e_n) = \frac{2^{-2M}}{3} \cdot \frac{1}{2\pi j} \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z}$$

$$+ \frac{2^{-2M}}{3}(1-2^{-2(N-1)}) \frac{1}{2\pi j} \oint_{|z|=1} \frac{1}{P(z).P(z^{-1})} .\frac{dz}{z}$$

(6.6.15)

for the two's complement rounding (discrete model).

Figure 6.6.2 shows the proposed generalized quantization noise model for the error analysis of a general second-order filter implemented by the proposed CBFCS approach.

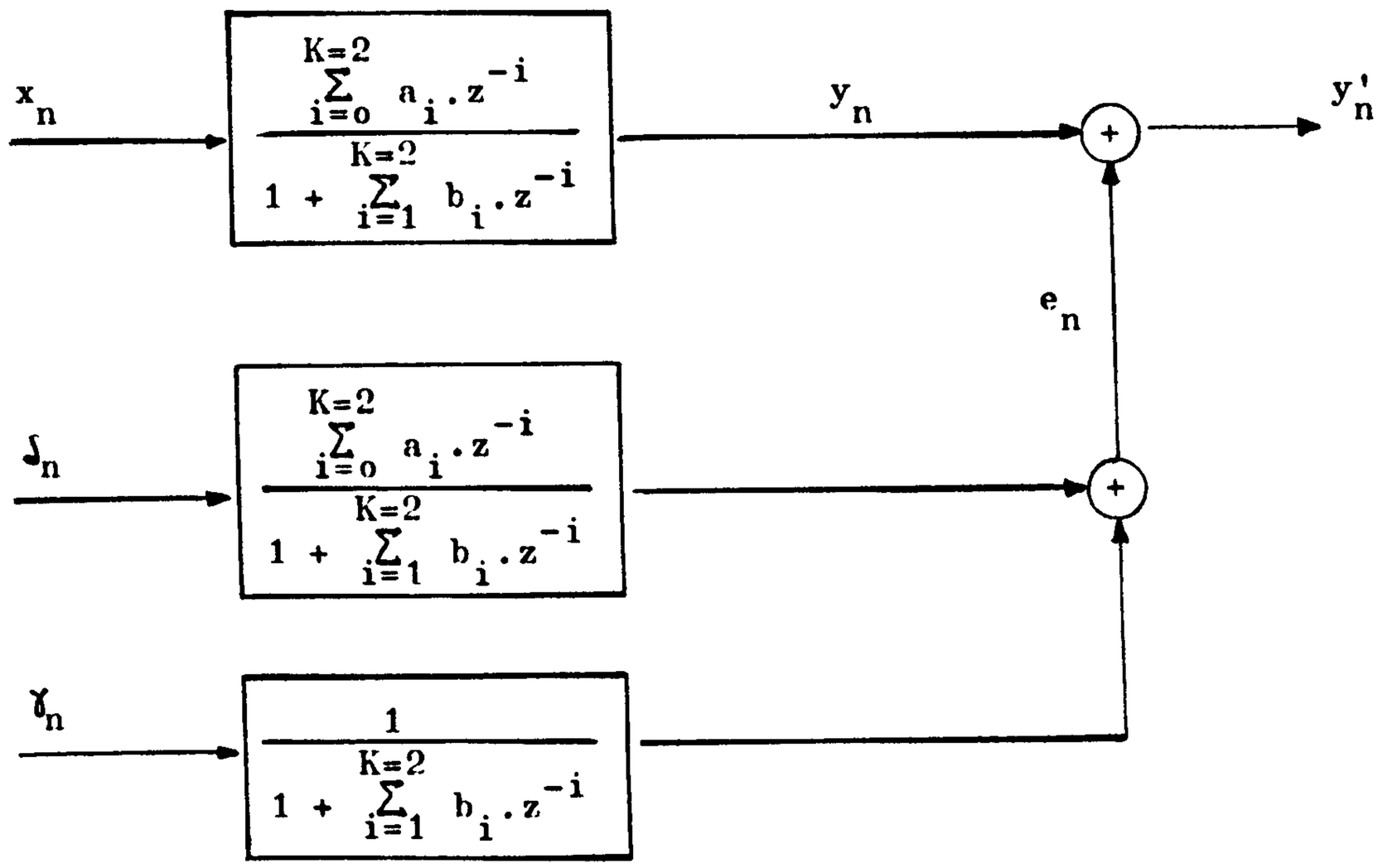


FIG: 6.6.2 A GENERALIZED QUANTIZATION NOISE MODEL FOR THE ERROR ANALYSIS OF A QUANTIZED GENERAL SECOND-ORDER FILTER IMPLEMENTED BY THE PROPOSED CBFCS APPROACH

Although in equation (6.6.1) we have chosen the serial-mode implementation of the CBFCS algorithm to illustrate the error δ_n, similar equations can be easily written down for the parallel-mode implementation, and the above generalized quantization noise model holds for all modes of implementations.

Furthermore, as a consequence of our previous assumption that H(z) is designed by the quantized design procedure discussed in section 6.4.2, we now have a resultant quantized filter which is asymptotically linear, that is, as the word length M of the filter variables increases, the filter response becomes closer to that of an ideal linear filter. This is, however, not the case if H(z) has

coefficients which are obtained by simply rounding or truncating those of a corresponding ideal transfer function represented with infinite precision, as the CBFCS function f in equation (6.6.1) is then subjected to coefficient quantization error.

If the digital filter is implemented using two's complement truncation arithmetic or up-rounding arithmetic (discrete model), there is an additional mean error $\mathcal{E}(e_n)$ which can easily be removed at the output of the filter by subtracting the value of $\mathcal{E}(e_n)$ given by equation (6.6.7) previously. In cases where the final digital output of the filter is to be reconstructed back to an analogue signal through a digital-to-analogue converter, the above mean error can then simply be removed by adding an appropriate d.c. bias at the output of the digital-to-analogue converter.

So far, our discussion of the proposed generalized quantization noise model has been based on a general second-order section. However, using the above model, the error analysis of a general Kth order digital filter implemented by the CBFCS approach is rather straightforward. Although a general Kth order can be directly implemented by the proposed CBFCS approach, the parallel form or the cascade form of implementation is generally preferred over the direct form for reasons of coefficient sensitivity. In the following we shall discuss the errors involved in the implementation of a Kth order digital filter by the various options viz., the direct form, the parallel form and the cascade form.

For the direct form of implementation, the digital filter has a transfer function:

$$H(z) = \frac{\sum_{i=0}^{K} a_i . z^{-i}}{1 + \sum_{i=1}^{K} b_i . z^{-i}} \qquad (6.6.16)$$

with a pole expression:

$$P(z) = 1 + \sum_{i=1}^{K} b_i . z^{-i} \qquad (6.6.17)$$

The error analysis of the Kth order digital filter is then analogous to that of the second-order section previously discussed, and can be performed by using figures 6.6.1 and 6.6.2 where K=2.

The variance $V(e_{D,n})$ of the output error in the direct form of implementation of a general Kth order digital filter, due to the input quantization error δ_n and the multiplication roundoff error γ_n, is simply obtained by substituting equations (6.6.16) and (6.6.17) into equations (6.6.14) and (6.6.15). Similarly, the mean error at the output can also be obtained by using equation (6.6.17) in equation (6.6.7) appropriately. In short, the multiplication roundoff error γ_n passes through the poles of H(z) only, whereas, the input quantization error δ_n passes through the poles and zeros of H(z) as in the case of the second-order section shown in figures 6.6.1 and 6.6.2 where K=2.

On the other hand, the parallel form of implementation calls for a combination of a number of second-order section (shown in figures 6.6.1 and 6.6.2) in parallel. The general Kth order filter thus implemented, has a transfer function:

$$H(z) = \sum_{i=1}^{K/2} H_i(z) \qquad (6.6.18)$$

where
$$H_i(z) = \frac{a_{1i}.z^{-1} + a_{oi}}{b_{2i}.z^{-2} + b_{1i}.z^{-1} + 1} \qquad (6.6.19)$$

represents the ith second-order section with a pole expression

$$P_i(z) = b_{2i}.z^{-2} + b_{1i}.z^{-1} + 1 \qquad (6.6.20)$$

Therefore, the total output error due to a parallel combination of the above K/2 second-order sections has a variance given by

$$V(e_{P,n}) = \sum_{i=1}^{K/2} V(e_{n,i}) \qquad (6.6.21)$$

where the variance, $V(e_{n,i})$ of the output error of the ith second-order section is obtained by substituting equations (6.6.19) and (6.6.20) in equations (6.6.14) and (6.6.15) of the proposed generalized quantization noise model. Hence, the error analysis of a Kth order digital filter implemented in the parallel form is

simply performed by paralleling K/2 second-order noise model each represented by figure 6.6.2 and the total output error variance is then the sum of the output noise variances of the individual sections (as given by equation (6.6.21) above). In equation (6.6.21) above, the summation of the individual output noise variances has been performed under the implicit assumption that these output errors are mutually independent of each other, that is uncorrelated. Moreover, equation (6.6.21) does not take into account the additional error incurred in the final scaling (if necessary) of the outputs of the individual second-order sections before summing them up. However, if the final summation is performed with a few extra bits, then the scaling error would become negligible. In any case, if the scaling coefficients are known, any scaling errors incurred can be easily accounted for.

The cascade form of implementation calls for a combination of K/2 second-order sections (each of which is implemented in the direct form as shown in figure 6.6.1) in cascade. The general kth order filter thus implemented, has a transfer function:

$$H(z) = \prod_{i=1}^{K/2} H_i(z) \qquad (6.6.22)$$

where
$$H_i(z) = \frac{a_{2i}.z^{-2} + a_{1i}.z^{-1} + a_{oi}}{b_{2i}.z^{-2} + b_{1i}.z^{-1} + 1} \qquad (6.6.23)$$

represents the ith second-order section with a pole expression

$$P_i(z) = b_{2i}.z^{-2} + b_{1i}.z^{-1} + 1 \qquad (6.6.24)$$

The total output variance of the cascade form is given by

$$V(e_{C,n}) = \frac{V(\delta_n)}{2\pi j}\oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z} + \sum_{i=1}^{K/2}\frac{V(\delta_n)}{2\pi j}\oint_{|z|=1}\frac{1}{P_{\frac{1}{2}K}(z).P_{\frac{1}{2}K}(z^{-1})}.\frac{dz}{z}$$
$$+ \sum_{i=1}^{K/2}\frac{V(\delta_n)}{2\pi j}\oint_{|z|=1}\frac{1}{P_i(z).P_i(z^{-1})}.H_{i,\frac{1}{2}K}(z).H_{i,\frac{1}{2}K}(z^{-1}).\frac{dz}{z} \qquad (6.6.25)$$

where $P_{\frac{1}{2}K}(z)$ is the pole expression of the $\frac{1}{2}K$th second-order section and $H_{i,\frac{1}{2}K}(z)$ is the overall transfer function of the cascade combination of a total of $(\frac{1}{2}K+1-i)$ second-order sections starting from the ith second-order section to the $\frac{1}{2}K$th second-order section. That is,

$$H_{i.\frac{1}{2}K}(z) = \prod_{i}^{K/2} H_i(z) \qquad (6.6.26)$$

Also, the values of the variances $V(\delta_n)$ and $V(\gamma_n)$ are given by

$$V(\delta_n) = \frac{2^{-2M}}{3} \qquad (6.6.27)$$

and

$$V(\gamma_n) = \begin{cases} \dfrac{2^{-2M}}{3} & \text{for the continuous model,} \\ \dfrac{2^{-2M}}{3}(1-2^{-2(N-1)}) & \text{for the discrete model.} \end{cases} \qquad (6.6.28)$$

To conclude this section, we like to point out that the input and output signal-to-noise ratios of the above quantizatior noise model can be similarly defined as in section 6.3.1. In the latter part of this chapter, we shall discuss the effects of quantization errors when they are no longer statistically uncorrelated.

6.7 DYNAMIC RANGE CONSTRAINTS IN THE PROPOSED HARDWARE IMPLEMENTATION

Every finite word length machine has a dynamic range associated with it. For instance, the number of binary digits (bits) used in the internal arithmetic operations of a digital filter may be thought of as determining either its quantization step size or its "dynamic range". As a direct consequence, the number of digits representing the internal signals of a digital filter limits the maximum allowable signal level and, hence, the maximum finite-word-length filter gain and realizable signal-to-noise ratio. In section 6.3, the inter-relationship of dynamic range and signal-to-noise ratio had been examined. In section 6.5, the interaction of dynamic range and multiplication roundoff noise had been established. While the proposed generalized quantization noise model sums up the effects

of a finite dynamic range on the resultant quantization noise in the implementation of a digital filter by the proposed CBFCS approach, this section is concerned with the more immediate problem of scaling often required to meet the dynamic constraints of a finite word length digital filter in general.

The fact that the digital filter has a finite internal word length gives rise to another phenomenon called "signal overflow" which occurs when the signal amplitude exceeds the dynamic range of the filter. For the proposed CBFCS implementation, the filter variables are represented by M-bit two's complement numbers such that the resultant dynamic range is

$$-1 \leqslant \text{filter variable} \leqslant (1-2^{-(M-1)}) \qquad (6.7.1)$$

When the input dynamic range is exceeded, we have the undesirable effects of the resultant input clipping. This can be equivalently viewed as the dynamic range of the input quantizer in figure 6.6.1 being exceeded and the input signal has then to be "properly" scaled (see section 6.3.1) if the maximum signal-to-noise ratio is to be maintained without exceeding the dynamic range constraints. On the other hand, if the output dynamic range is exceeded, a phenomenon called "overflow oscillations" will occur and the resultant filter is then non-linear. This can be equivalently viewed as the dynamic range of the output quantizer in figure 6.6.1 being exceeded. The problem can be approached in several ways.

The first approach to prevent output overflow is by further scaling the input to the filter by a factor equal to the gain of the filter. However, as pointed out in section 6.3.1, this amount of reduction in signal strength will correspondingly decrease the input signal-to-noise ratio, which is not desirable in most cases. On the other hand, the second approach relies on the fact that scaling the input signal by a factor equal to the filter gain is equivalent to normalizing the filter gain. This further implies the scaling of the filter gain $|H(e^{j\omega T})|$ to prevent output overflow. Furthermore, if $H'(z)$ is the scaled transfer function of $H(z)$, then the dynamic range constraints of the digital filter dictates that

$$\left|H'(e^{j\omega T})\right| \leq 1 \qquad (6.7.2)$$

and to preserve the largest possible signal-to-noise ratio, the equality in the above inequality (6.7.2) must be used such that the filter gain is exactly unity, that is, normalized. The third approach is to use a type of saturation arithmetic to quench the resultant "overflow oscillations" when "signal overflow" has occurred, such that the digital filter can resume its linear operations. Obviously this is an indirect or roundabout approach, for instead of directly preventing "signal overflow" to occur, one simply guarantees that the non-linear effects of "signal overflow" (that is "overflow oscillations") will not present in the resultant digital filter which employs a type of saturation arithmetic. The last approach and the phenomenon of "overflow oscillations" will be discussed in the next section, and it is suffice to say here that a mixture of the above methods is used in practice in coping with the problem of "signal overflow" due to a finite dynamic range.

At this juncture, it might be worth mentioning that the above problem of "signal overflow" seldom happens in systems which employ floating-point arithmetic. This is due to the much larger dynamic range of floating-point representation when compared with fixed-point representation for the same number of bits used. Moreover, floating-point arithmetic has an internal automatic scaling property as opposed to fixed-point arithmetic where the above necessary scaling has to be provided externally.

Although scaling is a solution to the above mentioned dynamic range problem of a finite word length digital filter, severe scaling, on the other hand, can sometimes cause another phenomenon called "signal underflow" to occur. This is the case when the signal strength falls short of the quantization step of the quantizer in question. Nevertheless, the occurrence of "signal underflow" is undesirable when a digital filter is used to process weak signals, for these weak signals within the digital filter may be wiped out completely as if they were never present at all. In these cases, the need of preventing "signal underflow" then forms a dynamic range constraint.

Finally, to conclude this section, we discuss the dynamic range constraints associated with the CBFCS memory which stores the pre-computed compensated partial products in the proposed CBFCS algorithm discussed in chapter five previously. These compensated partial products are pre-computed according to the CBFCS function f given in equation (5.2.10). Furthermore, in the formulation of the CBFCS function f, we have assumed, for convenience, that each of the above mentioned compensated partial products can be represented by an N-bit fraction, such that, the dynamic range of these compensated partial products is:

$$-1 \leq \text{compensated partial product} \leq (1-2^{-(N-1)}) \qquad (6.7.3)$$

This is equivalent to saying that the dynamic range of the CBFCS memory is equal to that given by equation (6.7.3) above.

However, these compensated partial products can assume a magnitude greater than unity, depending on the values of the filter coefficients which have also been assumed to be represented by N-bit fractions for convenience in equations (5.2.4) and (5.2.5). These filter coefficients can, in general, be greater than unity and will then be scaled to a magnitude less than unity. In both cases, a scaling of the CBFCS function f is implied so that the compensated partial products can be accommodated in the CBFCS memory without exceeding its dynamic range constraints. Effectively, this scaling of f corresponds to modifying equation (5.2.10) as below:

$$f_s(r_1^j, r_2^j, r_3^j, r_4^j, r_5^j, C_B^{j+1}, C_{2B}^{j+1}, \ldots\ldots, C_{\left(\frac{N}{B}-1\right)B}^{j+1})$$

$$= 2^{-m}(a_o.r_1^j + a_1.r_2^j + a_2.r_3^j - b_1.r_4^j - b_2.r_5^j)$$

$$+C_B^{j+1}.2^{-B} + C_{2B}^{j+1}.2^{-2B} + \ldots\ldots\ldots + C_{\left(\frac{N}{B}-1\right)B}^{j+1}.2^{-\left(\frac{N}{B}-1\right)B} \qquad (6.7.4)$$

Also, equation (5.2.11) will be accordingly modified by substituting f_s as given by equation (6.7.4) above in it in place of the unscaled function f. In the majority of cases, m=1 would be adequate. The above scaling will also be applied to the parallel implementation of

the proposed CBFCS algorithm and appropriate modifications of the equations thereof can be made similarly as shown above for the serial implementation.

Nevertheless, the above scaling of the CBFCS can be regarded as one possible way of accommodating non-fractional magnitude in the proposed CBFCS algorithm. On the other hand, one can always directly implement the proposed CBFCS algorithm using mixed-number representation by following the same approach outlined in chapter five. The corresponding error analysis of the mixed-number system can also be performed in a manner analogous to that discussed in the previous sections of this chapter. The necessary modifications of the equations derived so far are mathematically trivial and will, therefore, not be included here. In any case, whenever there is an advantage in using the above mixed-number representation in implementing a CBFCS digital filter, one should not hesitate to do so, for the above scaling of the CBFCS functions will then be avoided. As a consequence of using a mixed number representation, the CBFCS filter coefficients and variables can now take on a magnitude greater than unity and can be represented in two's complement notation as follows:

$$x_k = -x_k^o.2^S + \sum_{j=1}^{S} x_k^j.2^{S-j} + \sum_{j=S+1}^{M-1} x_k^j.2^{-(j-S)} \qquad (6.7.5)$$

$$y_k = -y_k^o.2^S + \sum_{j=1}^{S} y_k^j.2^{S-j} + \sum_{j=S+1}^{M-1} y_k^j.2^{-(j-S)} \qquad (6.7.6)$$

$$a_k = -a_k^o.2^T + \sum_{i=1}^{T} a_k^i.2^{T-i} + \sum_{i=T+1}^{N-1} a_k^i.2^{-(i-T)} \qquad (6.7.7)$$

$$b_k = -b_k^o.2^T + \sum_{i=1}^{T} b_k^i.2^{T-i} + \sum_{i=T+1}^{N-1} b_k^i.2^{-(i-T)} \qquad (6.7.8)$$

where S is the number of bits in the integer part of the filter variables and T is the number of bits in the integer part of the filter coefficients. Having considered the quantization noise effects in the previous sections for the case of uncorrelated noise, we shall further examine the effects of quantization in the implementation of the CBFCS algorithm when the quantization errors become correlated, in the next section.

6.8 LIMIT CYCLE BEHAVIOUR IN THE PROPOSED HARDWARE IMPLEMENTATION:

Sustained oscillations of various amplitudes, or limit cycles as they are generally referred to, can appear at the output of a finite word length digital filter even when there is no applied input, that is, under zero input conditions (104), (105), (106). The problem of limit cycles, however, can usually be ignored in digital filters which employ floating-point arithmetic because of the inherent automatic scaling property of the arithmetic (107). On the other hand, digital filters which employ fixed-point arithmetic exhibit two distinct types of oscillations.

The first type is essentially due to the overflow of the finite dynamic range registers in digital filters, as pointed out in the last section, and has been referred to as "overflow oscillations". These "overflow oscillations" are generally very large and undesirable, and hence, it is important to eliminate the possibility that they can occur. The second type of limit cycle in recursive digital filters results from amplitude quantization within the filters or, in particular, from the rounding or truncation of multiplication products in a feedback loop (108), (109), such that the resultant filters are non-linear. Such limit cycles were first noticed by Blackman (110), who referred to them as "deadband effects" and to the amplitude intervals within which these limit cycles are confined as "deadbands". At this juncture, it ought to be pointed out that limit cycles are inherent in recursive digital filters rather than non-recursive digital filters since these filters have no feedback loops in their structures. Moreover, if the amplitude of a limit cycle existing in a digital filter is large compared with its dynamic range, then the existence of such a limit cycle can limit the application of the digital filter in general (106), (111). This is particularly true in systems where idle channel noise must be minimized. We shall consider next, the second type of limit cycle in the proposed CBFCS approach.

In our previous discussion of quantization noise in digital filters implemented by the proposed CBFCS approach, we assumed that sample-to-sample changes in the input signal were large compared with the size of a quantization step, and that multiplication roundoff

errors were uncorrelated. However, when the input signal is a constant, the rounding errors resulting from fixed-point finite word length arithmetic can no longer be modeled as uncorrelated random variables. Moreover, as mentioned earlier, this is also the case for zero-input conditions. Hence, in the following analysis of limit cycles in the proposed CBFCS approach, we allow for the fact that the above rounding errors can be correlated.

Since high-order digital filters can be readily implemented by a parallel or cascade combination of second-order sections, we shall restrict our analysis of limit cycles in the proposed CBFCS approach to the basic second-order section shown in figure 6.6.1 in section 6.6. Furthermore, as the zero-input case is of more interest in practice, we shall consider only the zero-input case in deriving a bound for the possible limit cycles in a second-order section, although results of the same general character can be easily and similarly derived for any constant input.

In general, when a second-order section as shown in figure 6.6.1 is excited by a constant input, there may be many possible steady-state outputs confined within a "deadband", as the system is now considered as a non-linear feedback digital filter with errors that are correlated. However, a necessary condition for a steady-state constant output error when the input is a constant can be easily derived by a direct reference to figure 6.6.1 and as follows. Let the constant input be X and suppose the output has reached a steady-state value Y', not necessarily equal to the predicted value Y, which would occur if infinite precision arithmetic was used. Then from equations (6.6.1) and (6.6.6), we have

$$Y' = \left[\sum_{i=o}^{K=2} a_i.X - \sum_{i=1}^{K=2} b_i.Y'\right] + E \qquad (6.8.1)$$

where the error, E, for rounding, satisfies the inequality

$$-2^{-M} < E \leqslant 2^{-M} \qquad (6.8.2)$$

which is seen to be the roundoff error range of the output quantizer. Furthermore, equations (6.8.1) and (6.8.2) now permit us to write

an inequality for Y' as follows:

$$\left| (1 + \sum_{i=1}^{K=2} b_i)(Y' - \frac{\sum_{i=o}^{K=2} a_i}{1 + \sum_{i=1}^{K=2} b_i} X) \right| \leqslant 2^{-M} \qquad (6.8.3)$$

since the magnitude of the error E is bounded by the inequality

$$|E| \leqslant 2^{-M} \qquad (6.8.4)$$

Now, the value Y, which is computed with infinite precision, is the predicted value below:

$$Y = \frac{\sum_{i=o}^{K=2} a_i}{1 + \sum_{i=1}^{K=2} b_i} . X \qquad (6.8.5)$$

so that the deviation of the quantized steady-state output Y' from the predicted infinite precision value Y satisfies the inequality:

$$|Y' - Y| \leqslant \frac{2^{-M}}{\left| 1 + \sum_{i=1}^{K=2} b_i \right|} \qquad (6.8.6)$$

Therefore, inequality (6.8.6) implies that a necessary condition for a steady-state constant output Y' is that the value of Y' must satisfy the following inequality:

$$Y - \frac{2^{-M}}{1 + \sum_{i=1}^{K=2} b_i} < Y' \leqslant Y + \frac{2^{-M}}{1 + \sum_{i=1}^{K=2} b_i} \qquad (6.8.7)$$

The interval in the above inequality (6.8.7) is referred to as the deadband of the above second-order section. Furthermore, the above necessary condition is also a sufficient condition when the past (or delayed) two outputs of the second-order section are equal and lie within the deadband. The width of the above deadband is inversely proportional to the pole expression P(z) of the filter evaluated at

the point z=1 (the pole expression P(z) is given by equation (6.6.17) in section 6.6). The total number of quantization steps, each of width $2^{-(M-1)}$, in this deadband is approximately equal to the d.c. gain of the filter with poles only.

Furthermore, it is noticed in equations (6.8.6) and (6.8.7) that it is always possible to reduce the width of the above deadband by increasing the value of M. This implies the use of more bits to increase the finite dynamic range of the digital filter, thereby, decreasing the width of the quantization step. In addition, if truncation was used instead of rounding, the width of the resultant output deadband would be the same, but would not be symmetrically distributed about Y, as the inequality (6.8.2) would then be changed to a corresponding range for truncation. Nevertheless, the zero-frequency limit cycle associated with the above deadband is an exceptional case, whereas, in the general case, limit cycles can assume non-zero frequencies. We shall next derive a bound on limit cycles caused by roundoff errors in the basic second-order section shown in figure 6.6.1 in section 6.6, under zero-input conditions. In this instance, limit cycles are defined to be the autonomous d.c. or periodic behaviour of a digital filter under zero-input conditions after having been excited by a non-zero input.

Since a second-order digital filter implemented by the CBFCS approach is always in the direct, as pointed out previously, the output noise passes through the poles P(z) of the filter only. Thus, the zeros of the transfer function H(z) do not play any role in the analysis of zero-input limit cycles. Consequently, the basic second-order section shown in figure 6.6.1 can now be modified to become a pole-only structure, as shown in figure 6.8.1, which is described by

$$y'_n = e_n - \sum_{i=1}^{K=2} b_i . y'_{n-i} \qquad (6.8.8)$$

where the roundoff error term e_n in the above quantized recursive difference equation satisfies the inequality:

$$|e_n| \leqslant 2^{-M} \qquad (6.8.9)$$

Furthermore, e_n is defined as the total accumulation of roundoff

errors, and we assume that no overflows will occur.

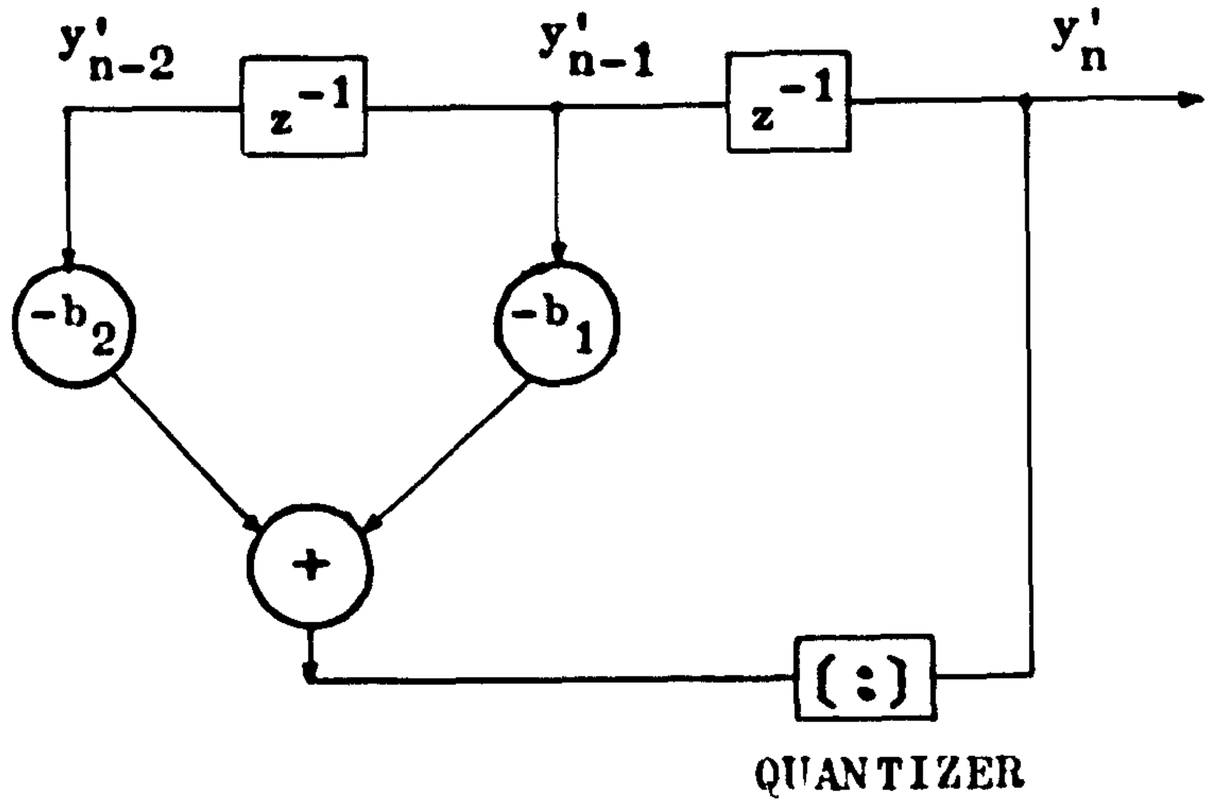


FIG: 6.8.1 BLOCK DIAGRAM OF A GENERAL SECOND-ORDER QUANTIZED CBFCS FILTER FOR THE ANALYSIS OF LIMIT CYCLES UNDER ZERO-INPUT CONDITIONS

A variety of bounds on limit cycles caused by roundoff errors have been developed (112), (113), (114), (115), and, in particular, we shall follow the approach of the latter, since it leads to an absolute bound on the peak value of any limit cycle with period L. Furthermore, we assume that the basic second-order section H(z) is asymptotically stable, that is, H(z) has all its poles P(z) strictly inside the unit circle. Hence, the region of stability which we restrict our analysis of limit cycles in, is the familiar triangle with vertices at the points (2, 1), (-2, 1) and (0, -1) in a b_1-b_2 plane (129). Furthermore, y'_n in equation (6.8.8) above must be bounded for any bounded input due to the above stability assumption. Taking the Z Transform of equation (6.8.8) we see that

$$Y'(z) = E(z)/P(z)$$

where E(z) is the Z Transform of the error e_n.

Now, if h_n is the impulse response of the pole expression P(z), then y'_n can theoretically be calculated by the convolution sum:

$$y'_n = \sum_{k=o}^{\infty} h_k \cdot e_{n-k} \qquad (6.8.10)$$

Since finite word length digital filters are finite-state machines, periodic limit cycles (including d.c. or zero-frequnecy limit cycles) must be reached in a finite number of steps, under zero-input conditions, because of the finite number of internal states of the filters (113). Consequently, we assume next that a limit cycle of period L exists, such that

$$e_n = e_{n+L} \qquad (6.8.11)$$

Similarly, it can be shown that

$$y'_n = y'_{n+L} \qquad (6.8.12)$$

and

$$\sum_{k=n+L}^{n+L} y'_k = 0 \qquad (6.8.13)$$

since the basic second-order section is assumed to be stable such that $(1 + b_1 + b_2) \neq 0$. Using the periodicity of e_n, equation ((6.8.10) can be written as

$$y'_n = \sum_{r=o}^{L-1} e_{n-r} \cdot \sum_{k=o}^{\infty} h_{r+kL} \qquad (6.8.14)$$

Therefore, by equation (6.8.9), we have

$$|y'_n| = 2^{-M} \cdot \sum_{r=o}^{L-1} \left| \sum_{k=o}^{\infty} h_{r+kL} \right| \qquad (6.8.15)$$

which provides an absolute bound on the peak value of any limit cycle with period L of the basic second-order section implemented by the proposed CBFCS algorithm.

In general, the limit cycle periods for different initial conditions cannot be determined except by exact simulation. For design purposes, a bound independent of L and n can be determined by observing from equation (6.8.10) or (6.8.15) that

$$|y'_n| \leqslant 2^{-M} \cdot \sum_{k=o}^{\infty} |h_k| \qquad (6.8.16)$$

Utilizing the asymptotic stability assumed on h_r, it can be shown that

$$\lim_{L\to\infty} \sum_{r=0}^{L-1} \left| \sum_{k=0}^{\infty} h_{r+kL} \right| = \sum_{r=0}^{\infty} |h_r| \qquad (6.8.17)$$

From equation (6.8.15), it is seen that the absolute bound on the magnitude of any limit cycle of period L reduces in value with an increased value of M (the number of bits used for the dynamic range). This implies, as previously mentioned in the constant input case for a zero-frequency limit cycle (L=1), the use of more bits to increase the dynamic range of the digital filter, thereby, decreasing the width of the quantization step size $2^{-(M-1)}$.

The extension of the above bounds derived in equations (6.8.15) and (6.8.16) to higher-order filters implemented as either a parallel or cascade combination of basic second-order sections is obvious in the former case, but not in the latter. In the parallel case, the individual second-order sections share a common input, and their outputs are summed together to produce the filter output. Hence, the limit cycle behaviour of each section is independent of all other sections, and the output of the filter for zero-input must be simply the sum of the limit-cycle outputs of the individual second-order sections. In the cascade case, however, each second-order section derives its input from the preceding section, and thus, in general, only the first second-order section in the cascade combination has necessarily zero input when the filter is under zero-input conditions. It has been speculated (113) that the cascade filter behaves as if each second-order section were producing its own maximum amplitude limit cycle or else is filtering the limit cycle from the previous stage, whichever is the larger output. Furthermore, it has been found (116)-(120) that the ordering of poles and zeros in a cascade combination of second-order filters to realize a higher-order filter has a profound effect in minimizing roundoff noise which certainly has its effect on the resultant limit cycle behaviour of the cascade filter. It has been found (120) that peak noise can be minimized by arranging second-order sections approximately in the order of decreasing Q-factors. In this case, the high-Q second-order section which has the largest peak response and hence will amplify the

roundoff noise most, is placed early in the cascade filter such that it encounters the maximum filtering actions. However, this configuration requires the maximum amount of prescaling of the input signal to the digital filter to prevent overflow. On the other hand, in the somewhat more common case of desiring to minimize the total noise energy, the second-order sections are then arranged in ascending order of their Q-factors.

Having considered bounds for the limit cycles in the proposed CBFCS implementation, we shall now consider some remedies to the problem of limit cycles in general. It has been found (110) that the deadband effect of a digital filter can sometimes be avoided by adding a small noise to the input to the filter, and the process is known as "dithering". For example, one general form of a dither signal is

$$(-1)^n.2^{-M} \qquad (6.8.18)$$

which is periodic and has an amplitude 2^{-M} (i.e. half the width of the quantization step size $2^{-(M-1)}$). Other forms of dither signals of larger amplitudes are often used in practice to break up limit cycle behaviours. That a dither signal within a digital filter can sometimes be very effective in breaking up its limit cycle behaviour, is due to the fact that the dither signal helps the output of the filter to jump across its quantization thresholds. Furthermore, random noise of one or more bits in an input signal can also have the effect of a dither signal. (It had been found experimentally (111) that one bit of random noise in the input signal had broken up the limit cycle behaviour of a cascade filter which had been previously triggered into a limit cycle). Furthermore, this process of injecting random noise forces the resultant roundoff errors within a digital filter to become random. In particular, this iden has been applied to the rounding process in a digital filter and has been referred to as "random rounding" (121), (122). However, the randomness of the above roundoff errors causes a non-zero output signal even for a zero input. Consequently, instead of a true zero output for a zero input, a noise signal, known as "remaining noise" (121), of a relatively small amplitude is produced at the output. In addition, the above process of "random rounding" also introduces an increased quantization error

when compared with ordinary rounding (121), (123). However, it is also found (121), (123) that the use of "random rounding" can eliminate the limit cycle behaviour of a digital filter. Besides "random rounding", it is also worth mentioning that there exist some criteria fot the absence of limit cycles in a digital filter under zero-input conditions (124)-(126), however, these criteria are generally quite restrictive in the design of high-gain filters.

In the experimental verification of the proposed CBFCS hardware implementation of digital filters and the investigation of the proposed generalized quantization noise model for its error analysis, we had also tested the above principle of "random rounding" (122), (123) in addition to the usual two's complement up-rounding scheme described in section 6.5. In particular, it was found experimentally that by selectively forbidding some output states of a finite word length digital filter, in addition to applying the above principle of "random rounding", the limit cycle behaviour of a second-order filter had been eliminated such that a true zero output was obtained under zero-input conditions, in contrast to the non-zero noise-like output of the above "random rounding" scheme. Consequently, we have called this rounding process the "selective rounding scheme" as opposed to the above mentioned "random rounding" which does not require the absence of some selected output states of a finite word length filter. However, as has been found in the case of "random rounding" (121), (123), this proposed "selective rounding scheme" also introduces an inevitable increased quantization noise (due to the random noise injected) such that the scheme should only be used in filtering systems where the presence of limit cycles are undesirable. Such is the case for systems where idle channel noise is required to be a minimum.

In general, provided that the amplitude of a resultant limit cycle within a digital filter is not large compared with its dynamic range, the limit cycle behaviour of the filter may not limit its applications. The bounds of the limit cycles of digital filters implemented by the proposed CBFCS approach derived previously then give an indication of the "severeness" of the resultant limit cycles and the subsequent choice of rounding scheme to be used in a given application.

6.8.1 OVERFLOW OSCILLATIONS AND THE USE OF SATURATION ARITHMETIC:

Overflows occur in the proposed CBFCS implementation when the dynamic range of the "Adderless-Multiplierless Unit" (AMU) is exceeded. This equivalent to saying that when the total input to the output adder in figure 6.6.1 is out of the range:

$$-1 \leqslant \text{dynamic range of AMU} \leqslant (1-2^{-(N-1)}) \qquad (6.8.19)$$

(assuming pre-rounding is adopted at the start of the summation process). As already mentioned in section 6.7, the result of such overflows is the undesirable presence of "overflow oscillations", and the resultant filter then operates non-linearly (127)-(129). These "overflow oscillations" have been studied in details in the above references and will not be repeated here. On the other hand, the necessary and sufficient conditions for the absence of these "overflow oscillations" in a second-order recursive digital filter have been advanced in reference (130). It has been found and proved (127)-(129), (131) that the use of a type of saturation arithmetic

$$y'_n + V(-a_0 \cdot x'_n - a_1 \cdot x'_{n-1} - a_2 \cdot x'_{n-2} + b_1 \cdot y'_{n-1} - b_2 \cdot y'_{n-2}) = 0 \qquad (6.8.20)$$

will eliminate these "overflow oscillations" even under zero-input conditions. This is in fact equivalent to modifying the two's complement arithmetic in the CBFCS implementation to become

$$V(u) = \text{sgn}(u) \quad \text{for } |u|>1 \qquad (6.8.21)$$

whenever overflows occur, and when under normal two's complement arithmetic operation, we have,

$$V(u) = u \qquad \text{for } |u|\leqslant 1 \qquad (6.8.22)$$

The total input-output characteristic of the resultant modified "Adderless-Multiplierless Unit" becomes as shown in figure 6.8.2.

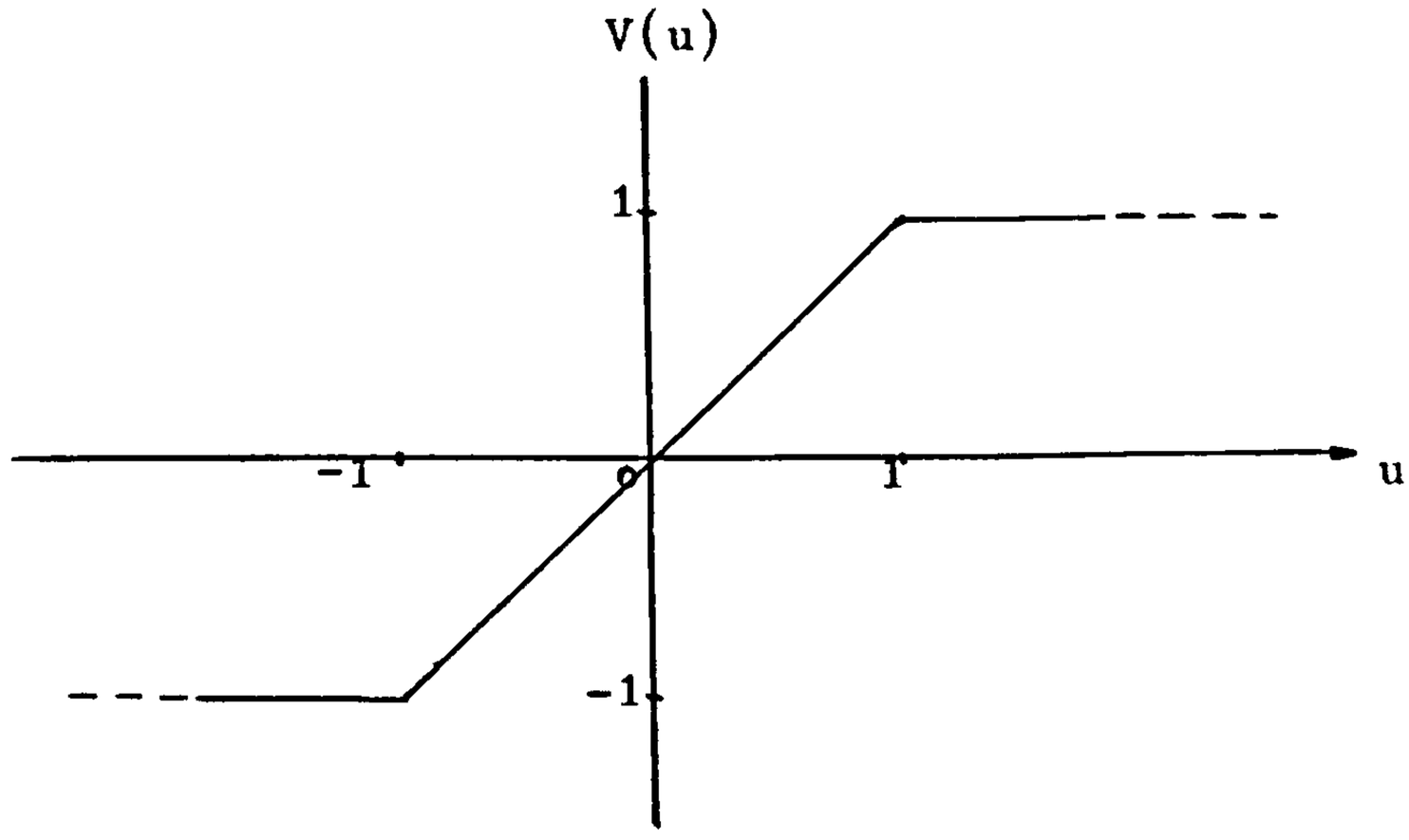


FIG: 6.8.2 MODIFIED TWO'S COMPLEMENT ARITHMETIC
FOR THE PROPOSED CBFCS IMPLEMENTATION

Under the input-output characteristic shown above, the value of the output sample y'_n will be assigned the value of $(1-2^{-(M-1)})$ in the event of a positive overflow in the summation of the partial products. In the event of a negative overflow, y'_n will be assigned the value of -1 instead. The use of this simple non-linearity guarantees stable behaviour whenever overflows occur. Of course, the filter is operating non-linearly at these times, however, it can be argued heuristically that saturation is perhaps the most sensible thing to do, as a remedy, in the case of overflow anyhow.

One comment is now in order. One only employs saturation arithmetic as a guarantee that "overflow oscillations" cannot occur. In general, the scaling rules in section 6.7 should be carefully observed such that a filter would be so designed that overflows are unlikely to occur during most of the processing time under normal operation conditions.

6.9 CONCLUSION:

The adoption of two's complement notation for signed filter coefficients and variables in the proposed CBFCS algorithm has made redundant the necessary code-conversion time required in other hardware implementations of digital filters where mixed

notations are used, as pointed out in section 6.2. Consequently, this represents a speed advantage of the proposed approach over other approaches employing mixed notations. Furthermore, the amount of time so gained can be turned into further useful signal processing.

From equation (6.3.9) in section 6.3, it is noticed that the output noise variance due to input quantization error is, in a sense, a time-dependent result since it is a function of n where n is the number of iterations. Furthermore, since the value of $|h_k^2|$ in equation (6.3.9) must be positive, we see that the output noise variance must increase with n from some originally minimum value. This is reasonable since one would not expect a large noise variance in the output of a digital filter immediately after noise is applied at its input. Physically, the variance of the output noise builds up and reaches an asymptote much as does a step signal applied to a linear discrete-time system. Eventually, a steady-state is reached unless the poles of the digital filter lie exactly on the unit circle. This is the case for the frequency sampling design technique discussed in chapter two, as opposed to the proposed design technique discussed in chapter three where the poles of the elemental filters are within the unit circle. Equation (6.3.8) describes the steady-state behaviour of the output noise variance, and when given the transfer function of a digital filter, the complex integral can be readily evaluated.

Section 6.3.1 introduced the interrelationship between input quantization and signal-to-noise ratios of the proposed implementation of digital filters. Explicit expressions have also been derived for the input and output signal-to-noise ratios for a given word length used. In deriving these espressions, we have implicitly assumed that the digital filter in question has been realized with infinite precision. The effect of coefficient quantization in the proposed hardware implementation was dealt with in section 6.4 where a quantized filter design procedure was given. Furthermore, section 6.4.1 dealt with errors due to coefficient quantization in the proposed design technique discussed in chapter three.

Section 6.5 described a discrete roundoff noise model for product quantization in the proposed hardware implementation of digital filters as opposed to the continuous noise model used for

the analysis of the effect of input quantization. This is due to the fact that the statistical assumptions made in the continuous model are only met in the input quantization case, while the statistical assumptions made in the discrete model are more realistic and consistent when one comes to access the noise output due to internal arithmetic roundoff in a digital filter. The noise variance given by equation (6.5.12) is in fact the output noise variance due to product quantization in a non-recursive digital filter implemented by the proposed CBFCS approach discussed in chapter five. Section 6.6 then introduced a generalized quantization noise model for the error analysis of the proposed CBFCS approach in hardware implementations. A comparision of equation (6.5.12) with equation (6.6.12) shows that the output noise due to product quantization is larger in the case of recursive realizations than non-recursive realizations. This is as expected since there are no feedback loops in the case of non-recursive realization while the output errors of recursive filters are continuously fedback into the pole sections which tend to amplify the resultant noise output.

The above proposed generalized quantization noise model for the proposed CBFCS implementation differs from conventional models for other approaches of implementation of digital filters in that there is only one output quantizer associated with the proposed model as opposed to the usual one-quantizer-per-coefficient realization scheme for other approaches. This represents a tremendous reduction in product quantization noise, for example, in an Nth order filter realized by conventional approaches there are 2N+1 coefficient multipliers and when realized in the direct form, the output noise variance of the filter due to product quantization is thus 2N+1 times that of the same Nth order filter realized by the proposed CBFCS approach. Furthermore, the replacement of hardware multipliers by the CBFCS memory technique has enabled local modifications within the CBFCS memory for a particular filter section which often help in noise reduction and the accuracy of the filter performance. These local modifications are simply achievied by modifying certain pre-computed "compensated partial products" before storing them in the CBFCS memory in the proposed CBFCS hardware implementation.

Having briefly compared the roundoff noise performance of the proposed CBFCS implementation with other conventional approaches, we now compare its noise performance with that of the "Combinatorial Filter" proposed by B.Liu et. al. (76). In their paper, B.Liu et. al. have arrived at the following noise expression for their method of hardware implementation of digital filters:

$$V(e_n) = \frac{2^{-2M}}{3}.\frac{1}{2\pi j}\oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z} +$$

$$\frac{2^{-2M}}{3}.\frac{1}{2\pi j}\oint_{|z|=1} P(z).P(z^{-1}).\frac{dz}{z}$$

A direct comparision of this equation with equation (6.6.15) shows that the noise variance in the "Combinatorial Filter" is substantially larger than that for the proposed CBFCS hardware implementation. This is due to the fact that B.Liu et. al. realization suffers from coefficient quantization effect while the proposed quantized design discussed in section 6.4 has rendered the proposed CBFCS implementation free from coefficient quantization error. Finally, filters implemented via B.Liu et. al. approach will not be asymptotically linear since the outputs of the combinatorial filters are non-linear functions of their filter coefficients as shown in the error analysis of B.Liu et. al. paper (76). On the contrary, as explained in section 6.4, filters implemented by the proposed CBFCS approach are asymptotically linear.

Finally, section 6.7 dealt with the dynamic range constraints of the proposed hardware implementation while section 6.8 dealt with the problems of limit cycle behaviours and overflow oscillations. Since, in general, the periods of limit cycles cannot be determined except by exact simulation, therefore, for design purposes, an upper bound independent of the periods of limit cycles was obtained as represented by equation (6.8.16) for the maximum amplitudes of these limit cycles.

CHAPTER SEVEN

LOGIC DESIGN AND CONSTRUCTION OF A DEMONSTRATION PROCESSOR AND ITS INTERFACING WITH ITS ASSOCIATED PERIPHERAL UNITS

7.1 INTRODUCTION:

A demonstration processor and its associated peripheral units were built to verify experimentally the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) discussed in chapter five, and to investigate the proposed generalized quantization noise model for the error analysis of the hardware implementation of the above scheme. The word length of this demonstration processor was chosen to be sufficiently short to further investigate the adverse effects of using such a short word length on the proposed design technique discussed in chapter three. Furthermore, the demonstration processor built was a basic second-order section, in view of the fact that higher-order filters could then be formed by multiplexing this basic section. Consequently, we shall concentrate on describing the logic design and construction of the above demonstration processor in this chapter, while further details pertinent to its multiplexing to form higher-order filters, in addition to those mentioned in section 5.4 in chapter five, will be furnished wherever appropriate.

Prior to describing in detail the logic design and construction of the above demonstration processor, we shall first lay down its specifications as follows:

(1) The word length M of the filter variables was chosen to be 8 bits including sign, in two's complement notation, so as to accommodate bipolar signals. Hence, all internal signals within the processor were quantized to 7-bit accuracy in magnitude in all arithmetic operations.

(2) The word length N of the filter coefficients was chosen to be 16 bits including sign, in two's complement notation, in order to implement a variety of filter responses. The filter coefficients were therefore represented to 15-bit accuracy in magnitude in all arithmetic operations.

(3) Two's complement up-rounding was chosen in preference to two's complement truncation, because of its better properties.

(4) Schottky TTL Random Access Memory (RAM) was chosen to implement the CBFCS memory for its speed and relatively low cost compared with other high-speed technologies such as ECL and BCL etc. The choice and use of RAM's further made possible a programmable, variable-coefficient filter by selectively changing the content of the CBFCS memory.

(5) Standard 4-bit TTL adder chips were used to implement the "Adderless-Multiplierless Unit" (AMU) in the implementation of the "Carry Brought Forward Compensation Scheme".

(6) The serial-mode implementation of the "Carry Brought Forward Compensation Scheme" was chosen since it required a minimum of hardware, but on the other hand, demonstrated the maximum complexity of the control circuitry required in implementing the CBFCS algorithm.

(7) Since the access time of the Schottky TTL RAM's chosen was 25nsec typically, the carry propagation path was "sectioned" at the "sectioning point" C_8, as explained in section 5.3, to give a summation time of the above AMU as 23nsec typically. This was then felt to be the best arrangement in "sectioning" the worst case N-bit carry propagation path down to a maximum of B bits long, where B=8 in this case.

(8) All essential logic units to ensure correct operations of the demonstration processor are shown in figure 7.1.1, while the operation of the demonstration processor is tabulated in table 7.1.1.

The operation of the demonstration processor shown in figure 7.1.1 can be described as follows. The register R_A shown in figure 7.1.1 enables concurrent operations of memory access of the CBFCS memory and the "byte-sliced" summation of the compensated partial products of f_n as described in detail in section 5.3.2 previously. However, it has also been pointed out that the introduction of R_A has made the demonstration processor into a "pseudo-pipeline" system which thus requires 9 clock cycles to produce an output sample y_n. In the above 9 clock cycles, the filter variables shift serially out of shift registers:

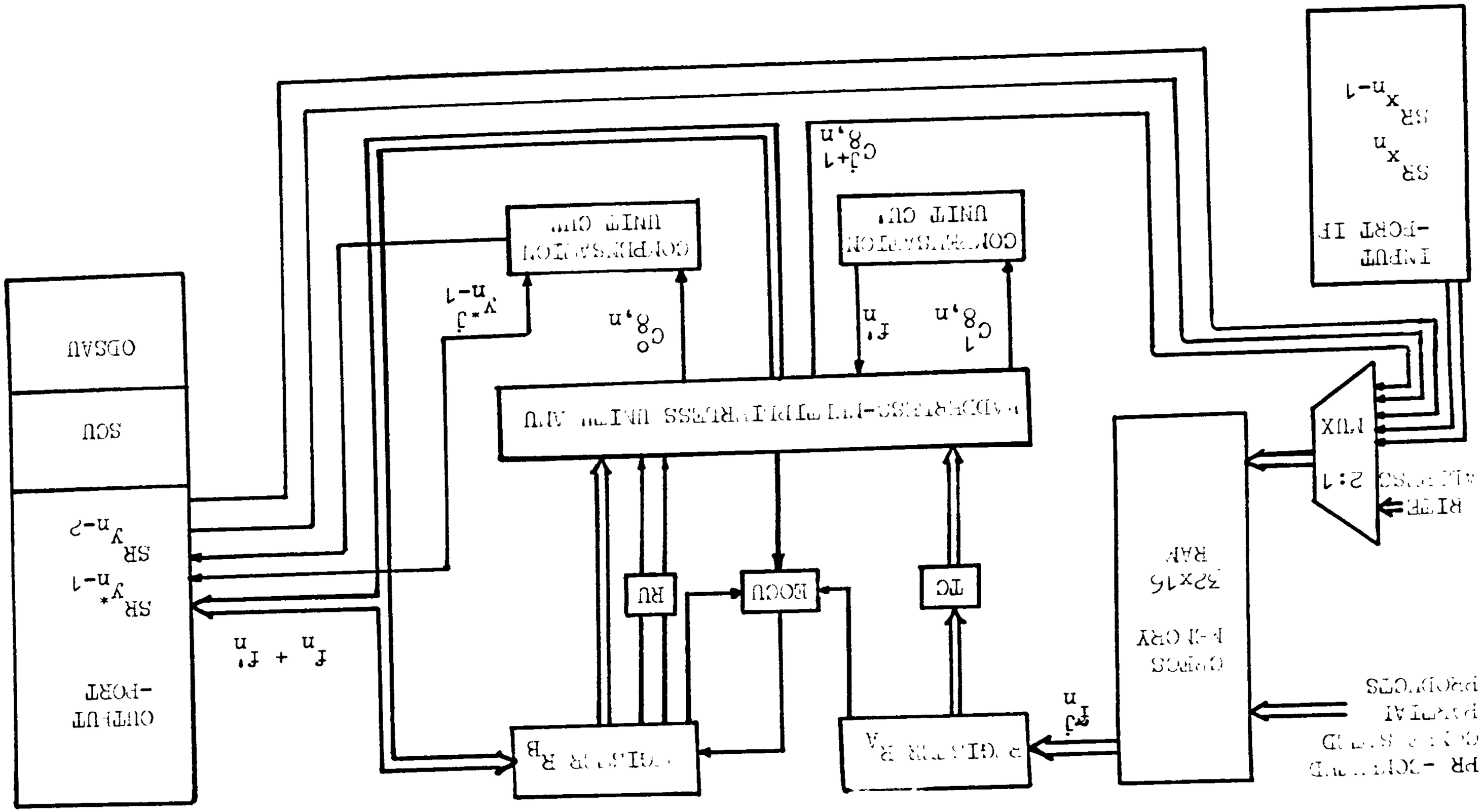
$$(SR_{x_n}, SR_{x_{n-1}}, SR_{y^*_{n-1}}, SR_{y_{n-2}})$$

At each shift, a new vector:

$$(x_n^j,\ x_{n-1}^j,\ y_{n-1}^j,\ y_{n-2}^j,\ C_{8,n}^{j+1})$$

appears at the address input of the CBFCS memory which realizes the function f as given in equation (5.2.10). The output of the CBFCS memory is $\tilde{f}_n^j$, the jth "compensated partial product" of f_n, which is then loaded in parallel into register R_A. The output of register R_A is connected to one of the two summation inputs of the "Adderless-Multiplierless Unit" (AMU). The other summation input of the AMU is hard-wired to the output of register R_B in parallel, with one bit right shift to weight the "depleted sum" of the AMU summation output. The AMU is made up of four standard TTL adder chips connected as logic function units, the details of which are shown in a later section. The carry output $C_{8,n}^{j+1}$ of the AMU is connected to both the compensation unit CU' and the compensation unit CU", such that its value is clocked into the appropriate units at the desired moment as described below. In addition, the carry output $C_{8,n}^{j+1}$ is used to address the CBFCS memory as mentioned above. The operation of the demonstration processor repeats every 9 clock cycles and in the first clock cycle, pre-rounding of the output sample is done via the rounding unit RU connected as shown in figure 7.1.1. This has the advantage of saving at least one clock cycle in the production of the output sample. Furthermore, in the first 8 clock cycles, the "compensated partial products" of f_n obtained from the CBFCS memory are summed in the proposed "byte-sliced" manner as described in section 5.2 previously. Any possible "end-overflows" during this time are corrected by the " End-Overflow Correction Unit" (EOCU) as shown in figure 7.1.1. Moreover, as pointed out in section 5.3.3, when right-shifting two's complement numbers, modifications have to be made in order to restore the proper weighting of the number being shifted. This is also performed by the above "End-Overflow Correction Unit" (EOCU) which thus serves a dual-purpose. In the 0th clock cycle, a "byte-sliced" subtraction is performed via the "True-Complementer" (TC) and a forced carry-in into the AMU. This is due to the negative weighting of the "compensated partial product" $\tilde{f}_n^0$. After 8 clock cycles, the value f_n' of the compensation function f' is

FIG:7.1.1 BLOCK DIAGRAM OF THE DEMONSTRATION PROCESSOR (MULTIPLEXING UNITS NOT SHOWN)



produced and clocked into the compensation unit CU'. The value of f_n is then first compensated by the value of f'_n in the 9th clock cycle when the above mentioned "byte-sliced" subtraction is performed, yielding the value $f_n+f'_n$ at the output of the AMU. The value f''_n of the compensation function f" is also produced at the end of the 9th clock cycle. While the value $f_n+f'_n$ is clocked in parallel into the shift register $SR_{y^*_{n-1}}$, the value f''_n is simultaneously clocked into the compensation unit CU" at the start of the 1st of the next nine clock cycles of the operation of the demonstration processor. At each data shift in the next nine clock cycles, the content of the shift register $SR_{y^*_{n-1}}$ is then shifted serially into the compensation unit CU" and is bit by bit compensated by the compensation function f" (see section 7.11) to yield the output

$$y^j_{n-1} = (f_{n-1} + f'_{n-1} + f''_{n-1})^j$$

which is the serial delayed version of the previous output sample. However, if the coefficients of the second-order difference equation being implemented have to be scaled in order to be accommodated in the CBFCS memory, the appropriate re-scaling of the above filter output has to be performed before the demonstration processor can be used to compute the next output sample. This is performed by the "Scaling Correction Unit" (SCU) shown together with the "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) and the shift registers

$$(SR_{y^*_{n-1}}, SR_{y_{n-2}})$$

as integral parts of the output port (OP) in figure 7.1.1. Similarly, the shift registers

$$(SR_{x_n}, SR_{x_{n-1}})$$

have also been shown in figure 7.1.1 as integral parts of the input port (IP). In the event of an overflow of the output of the demonstration processor, appropriate actions will then be taken by the above "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) to prevent overflow oscillations. Since, as mentioned in chapter six previously, the use of the proposed discrete two's complement up-rounding model requires the concurrent use of saturation

arithmetic, this "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) thus serves a dual-purpose.

CLOCK CYCLE	ADDRESS INPUT OF THE CBFCS MEMORY	OUTPUT OF THE "ADDERLESS-MULTIPLIERLESS UNIT" (AMU)
1	x_n^7, x_{n-1}^7, y_{n-1}^7, y_{n-2}^7, 0	r (PRE-ROUNDING BIAS SET)
2	x_n^6, x_{n-1}^6, y_{n-1}^6, y_{n-2}^6, $c_{8,n}^7$	$r.2^{-1} + \tilde{f}_n^7 - c_{8,n}^7.2^{-7}$
3	x_n^5, x_{n-1}^5, y_{n-1}^5, y_{n-2}^5, $c_{8,n}^6$	$r.2^{-2} + \sum_{j=6}^{7} \tilde{f}_n^j.2^{-(j-6)} - c_{8,n}^6.2^{-7}$
4	x_n^4, x_{n-1}^4, y_{n-1}^4, y_{n-2}^4, $c_{8,n}^5$	$r.2^{-3} + \sum_{j=5}^{7} \tilde{f}_n^j.2^{-(j-5)} - c_{8,n}^5.2^{-7}$
5	x_n^3, x_{n-1}^3, y_{n-1}^3, y_{n-2}^3, $c_{8,n}^4$	$r.2^{-4} + \sum_{j=4}^{7} \tilde{f}_n^j.2^{-(j-4)} - c_{8,n}^4.2^{-7}$
6	x_n^2, x_{n-1}^2, y_{n-1}^2, y_{n-2}^2, $c_{8,n}^3$	$r.2^{-5} + \sum_{j=3}^{7} \tilde{f}_n^j.2^{-(j-3)} - c_{8,n}^3.2^{-7}$
7	x_n^1, x_{n-1}^1, y_{n-1}^1, y_{n-2}^1, $c_{8,n}^2$	$r.2^{-6} + \sum_{j=2}^{7} \tilde{f}_n^j.2^{-(j-2)} - c_{8,n}^2.2^{-7}$
8	x_n^o, x_{n-1}^o, y_{n-1}^o, y_{n-2}^o, $c_{8,n}^1$	$r.2^{-7} + \sum_{j=1}^{7} \tilde{f}_n^j.2^{-(j-1)} - c_{8,n}^1.2^{-7}$
9	x_n^o, x_{n-1}^o, y_{n-1}^o, y_{n-2}^o, $c_{8,n}^o$	$r.2^{-8} + \sum_{j=1}^{7} \tilde{f}_n^j.2^{-(j-1)} - c_{8,n}^o.2^{-7} - f_n^o + f_n'$

TABLE 7.1.1 SERIAL OPERATION OF THE DEMONSTRATION PROCESSOR

It is further noticed that the sign bits of all filter variables are repeated in the 9th clock cycle due to the introduction of register R_A. Hence each data sample is padded with an extra bit to the right of the sign bit with a logical value same as that of the sign bit. In the next section, we shall discuss the control and timing circuitry which has been omitted in figure 7.1.1. Other logic units will be accordingly described in the latter part of this chapter which finishes with the interfacing of the demonstration processor.

7.2 THE CONTROL AND TIMING UNIT (CTU):

The main function of the control and timing unit (CTU) is to produce and distribute all essential timing and control signals and various instruction for the successful operation of all units. In essence, the instruction set for the operation of the demonstration processor shown in figure 7.1.1 includes the following:

(1) Loading and clearing of all registers.
(2) Presetting and clearing of all control flip-flops.
(3) Initialization of the demonstration processor.
(4) Generate all essential shift commands.
(5) Control of the "End-Overflow Correction Unit" (EOCU).
(6) Control of the "True-complementer" (TC).
(7) Control of the rounding unit (RU).
(8) Controls of the "Adderless-Multiplierless Unit" (AMU) and the compensation units CU' and CU".
(9) To furnish the enable, read and write signals of the CBFCS memory.
(10) Control of the "Scaling Correction Unit" (SCU).
(11) Control of the "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) and the initialization of saturation arithmetic.
(12) Control of data transfers and furnishing status flag signals.

The main control and timing circuitry is shown in figure 7.2.1 while other auxiliary control circuits will be shown in the relevant sections later. The control and timing waveforms generated are shown in figure 7.2.2.

Figure 7.2.1 shows the program counter which generates the control and status signals of the demonstration processor. When the demonstration processor is working at its maximum speed, a minimum propagation delay through its program counter is required. Using Schottky TTL technology, the above program counter can operate at a maximum clock rate of about 125MHz, with a typical propagation delay of approximately 4nsec. The status signals are obtained by decoding the outputs of the program counter. Note that the Q's and $\bar{Q}$'s of the program counter are obtained simultaneously. The decoding gates shown for the status signals can also be realized by NAND/NOR gates by some simple Boolean Logic manipulations of the gates' inputs. The decoded

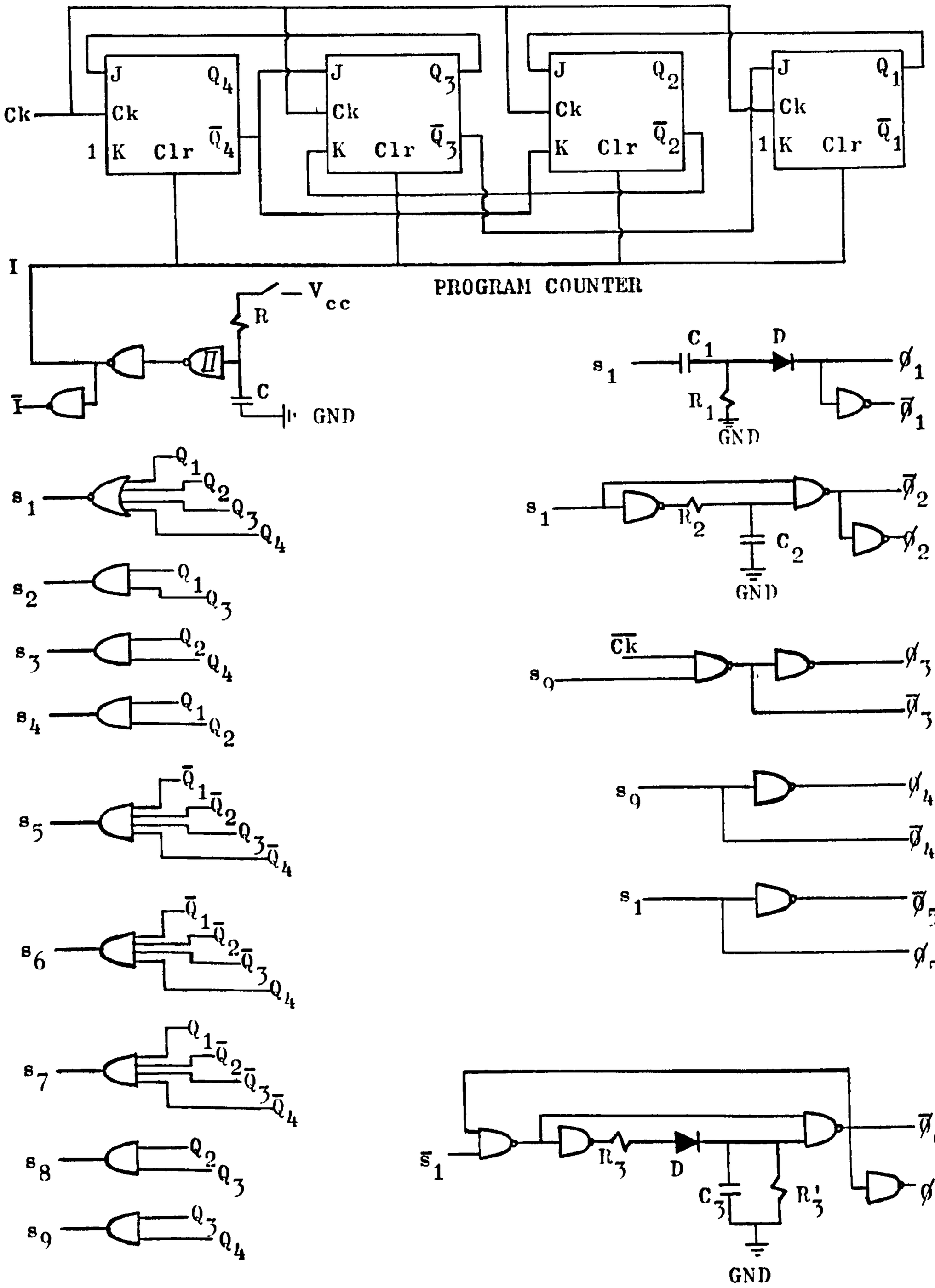
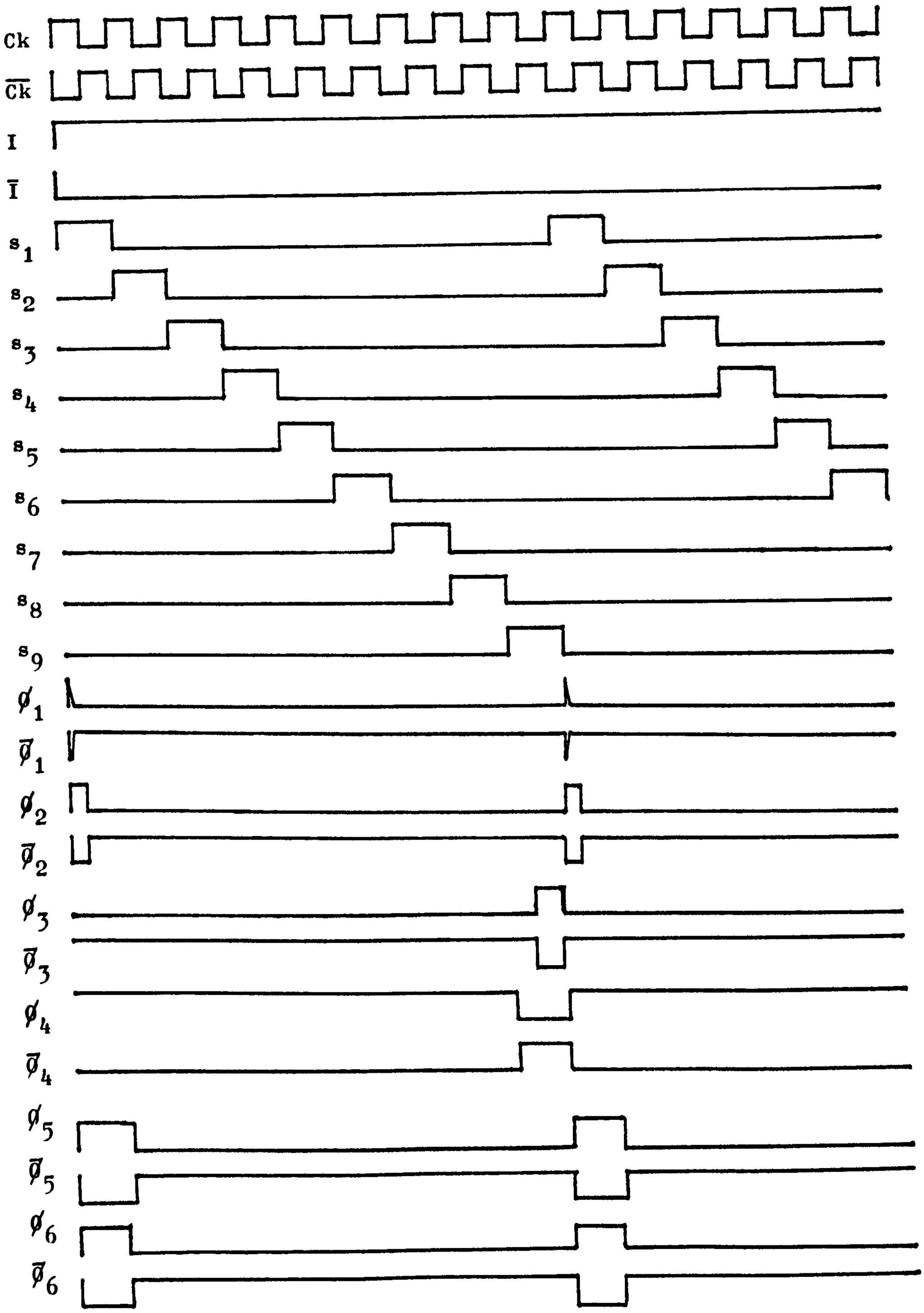


FIG: 7.2.1 CONTROL AND TIMING CIRCUITS



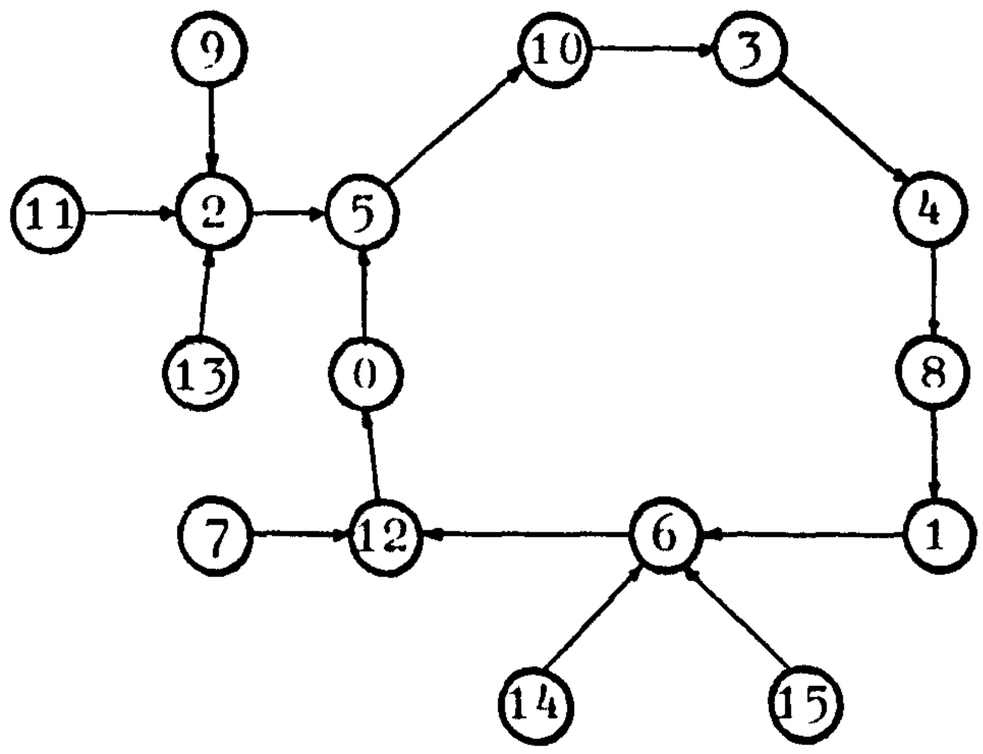
FIG; 7.2.2 CONTROL AND TIMING WAVEFORMS GENERATED BY CTU

status signals appear after one propagation gate-delay which is typically 3nsec using Schottky TTL technology. The "state diagram" and "state table" of the program counter shown in figure 7.2.1 are depicted below.

Q_4	Q_3	Q_2	Q_1	COUNT
0	0	0	0	1
0	1	0	1	2
1	0	1	0	3
0	0	1	1	4
0	1	0	0	5
1	0	0	0	6
0	0	0	1	7
0	1	1	0	8
1	1	0	0	9
0	0	0	0	10

REPEAT

STATE TABLE



STATE DIAGRAM

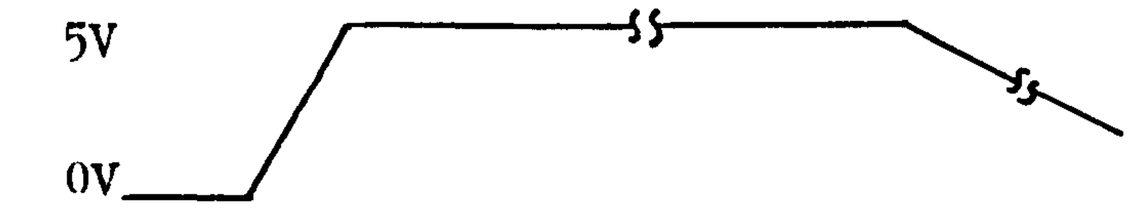
FIG: 7.2.3 STATE TABLE AND DIAGRAM OF THE PROGRAM COUNTER OF THE DEMONSTRATION PROCESSOR

The above state diagram shows how the states of the outputs of the program counter are "jumped" to yield a cycle length of 9, resulting in a "gateless" design as shown in figure 7.2.1. As a result, the program counter can be implemented by using two dual-JK integrated packages, while the inputs to these JK's require no external gating circuitry. Note that the design is synchronous so that the outputs of the program counter change states simultaneously at each clock pulse. This is important in high-speed operations to prevent decoding spikes which are hazardous and often result when decoding an asynchronous or ripple-through counter.

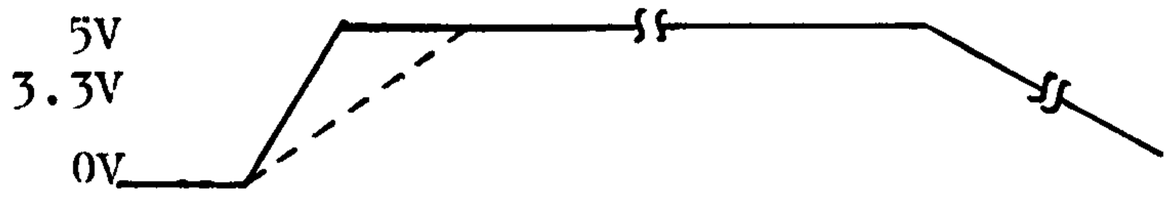
In applications where a longer propagation delay through the program counter can be tolerated or when the demonstration processor is not required to work at its maximum operational speed, integrated program counters can be used instead of the program counter shown in

figure 7.2.1. At this juncture, it is perhaps worth mentioning that the reason for designing the program counter of the demonstration processor out of JK flip-flops is that commercially available integrated program counters operate at a maximum clock frequency of about 40MHz and have much longer propagation delays. Furthermore, these integrated program counters only have the required Q outputs on chips, while their inverted values, the $\overline{Q}$ outputs are not available and have to be obtained externally via some inverters. Consequently, there exist uneven delays in obtaining the counter outputs and their inverted values. This may then cause decoding spikes in producing the control and status signals for the demonstration processor. However, these spikes can be masked by using long propagation delay gates, but then the overall propagation delay is much increased, making such integrated program counters suitable only for medium speed operations of the processor. In contrast, as mentioned previously, the program counter shown in figure 7.2.1 has both the Q and $\overline{Q}$ outputs seperately available so that they are obtained simultaneously without any uneven delays. Finally, standard off-the-shelf integrated program counters have preset counts so that a required count can only be obtained by modifying or programming, in the case of programmable counters, the outputs of these counters externally with some additional logic gates. On the other hand, the program counter in figure 7.2.1 does not require any external logic gates to provide the correct count, and is therefore referred to as "gateless".

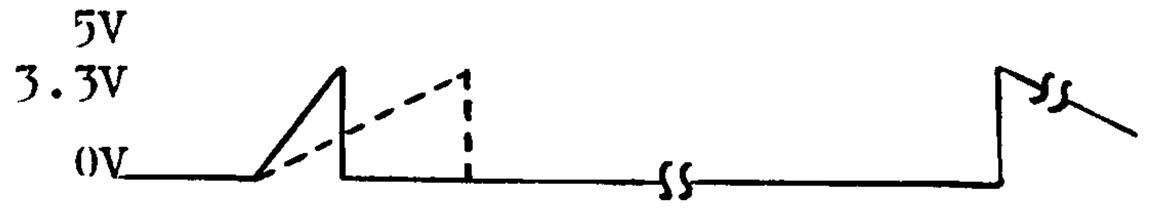
The initialization pulse I shown in figure 7.2.1 is used to initialize the operation of the demonstration processor when the power supply is first switched on. This pulse is produced by a technique based on the fact that different logic gates switch their output levels at different input thresholds. The circuit used is made up of a standard TTL NAND gate wired as an inverter, a Schmitt trigger gate and the RC integrating circuit shown. The input threshold of the standard TTL NAND gate is typically 1.5V while the Schmitt trigger switches between 0.9V to 1.7V typically due to its hysteresis property. The operation of the above initialization circuit is self-explanatory from figure 7.2.4 below.



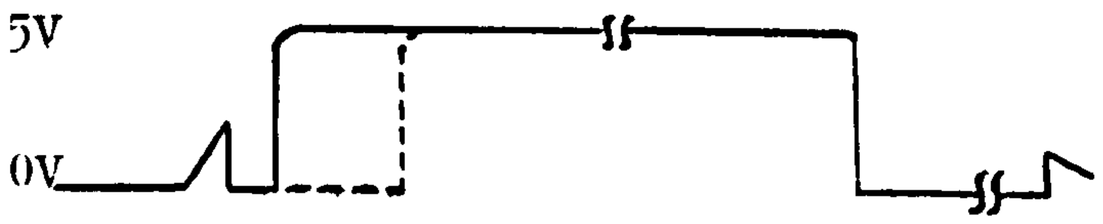
(a) WAVEFORM OF POWER SUPPLY



(b) INPUT WAVEFORM OF SCHMITT TRIGGER



(c) INPUT WAVEFORM OF STANDARD TTL GATE



(d) OUTPUT WAVEFORM OF STANDARD TTL GATE

FIG: 7.2.4 GENERATION OF AN INITIALIZATION PULSE I BY MEANS OF THE THRESHOLD DIFFERENCE BETWEEN A STANDARD TTL NAND GATE AN A SCHMITT TRIGGER.

The dotted lines above show the result of the introduction of the RC integrating circuit which is thus seen to ensure a certain minimum pulse width of the output initialization pulse. Hence, the time constant RC of the integrating circuit is so chosen to give a required pulse width. The above input waveform of the standard TTL NAND gate is in fact the output waveform of the Schmitt trigger gate which has a hysteresis characteristic shown below.

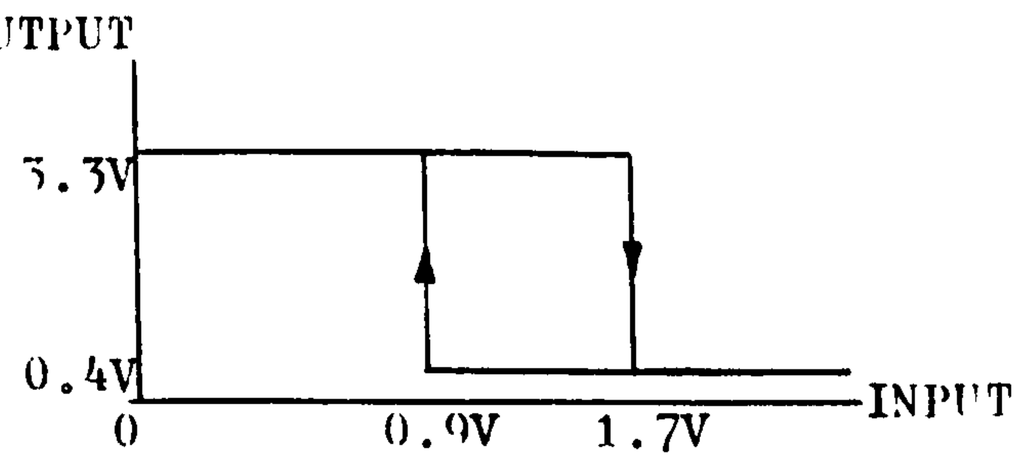


FIG: 7.2.5 HYSTERESIS OF A SCHMITT TRIGGER GATE

Hence, the standard TTL NAND gate changes state first due to its

lower operating threshold, while the Schmitt trigger changes state some time later due to its higher operating threshold. When the Schmitt changes state, the standard TTL NAND gate is thus forced to change state again, producing an output pulse whose duration depends on the time constant RC of the above integrating circuit which is necessary when the power supply has a fast rise time.

The generation of the control signals shown in figure 7.2.1 will now be briefly discussed before we introduce the problem of "clock skew" and synchronization of the control and timing signals with the clock signal of the demonstration processor. The control signal ϕ_1 is generated by differentiating the status signal s_1 by the R_1C_1 circuit. The diode D is used to block the negative swing of the differentiating circuit output. The time constant R_1C_1 is chosen to be much less than the width of the status pulse s_1 so that a narrow strobe pulse will result from differentiating s_1. The control signals ϕ_3, ϕ_4 and ϕ_5 are easily generated as shown in figure 7.2.1 and do not warrant further explanation. The generation of ϕ_2 is depicted in figure 7.2.6 below.

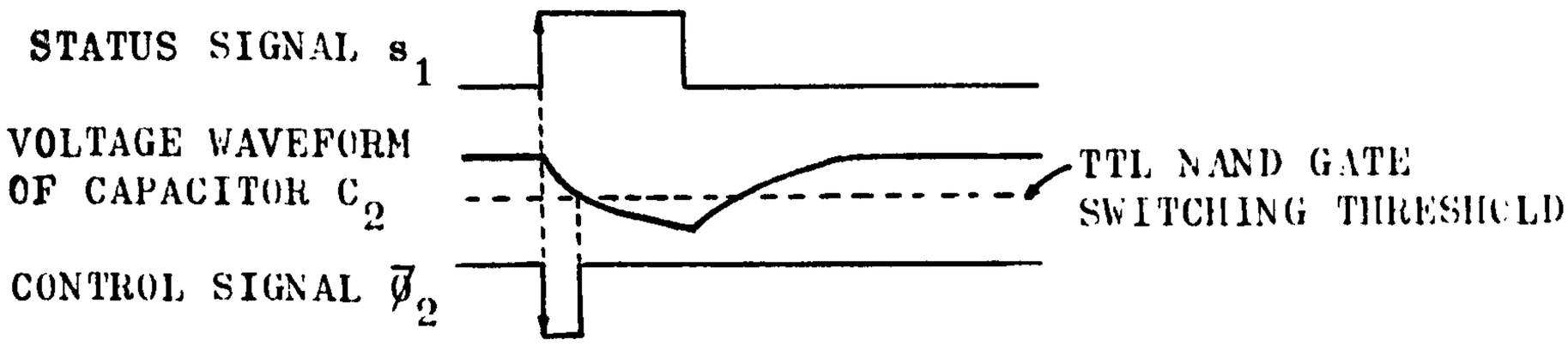


FIG: 7.2.6 PRODUCTION OF CONTROL SIGNAL ϕ_2

The time constant R_2C_2 of the integrating circuit should be chosen to be less than the width of s_1. In the duration of s_1, the capacitor C_2 discharges, while it charges up again when s_1 returns to logic 0. During the discharge of C_2, the voltage on C_2 crosses the input logic threshold of the output NAND gate and $\bar{\phi}_2$ is hence produced. The width of the control signal ϕ_2 is approximately equal to R_2C_2. The control signal ϕ_6 is generated in a similar manner, however, the introduction of the diode forces the capacitor C_3 to discharge through the resister R_3' in parallel with the input resistance of the output NAND gate, with a time constant equal to

$C_3.R'_3$. However, the charging time constant of the circuit is C_3R_3 and the choice of the time constants should be such that $C_3R_3 < C_3R'_3$. Note that this circuit can also produce a control signal with a pulse width wider than that of s_1 from which it is derived. The circuit operation is shown in figure 7.2.7 below.

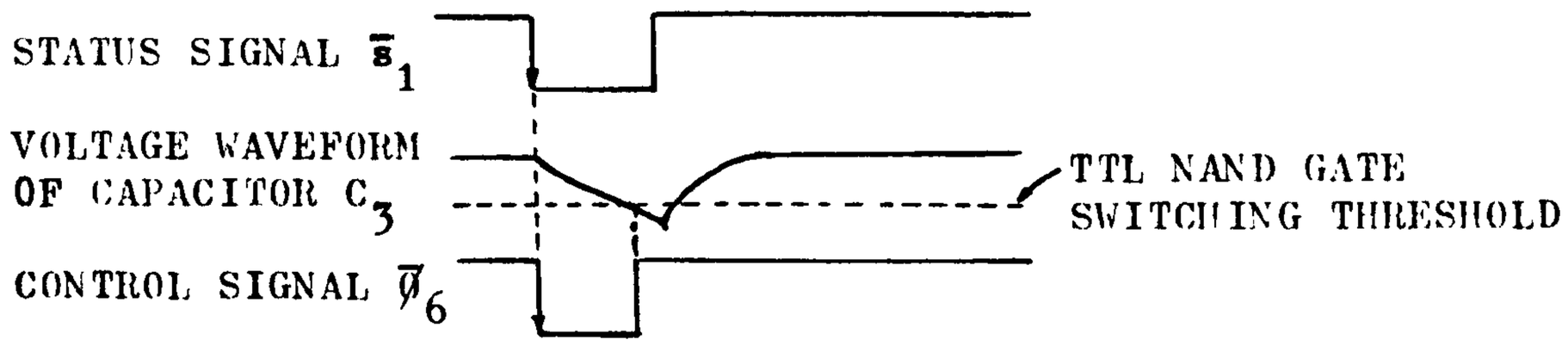
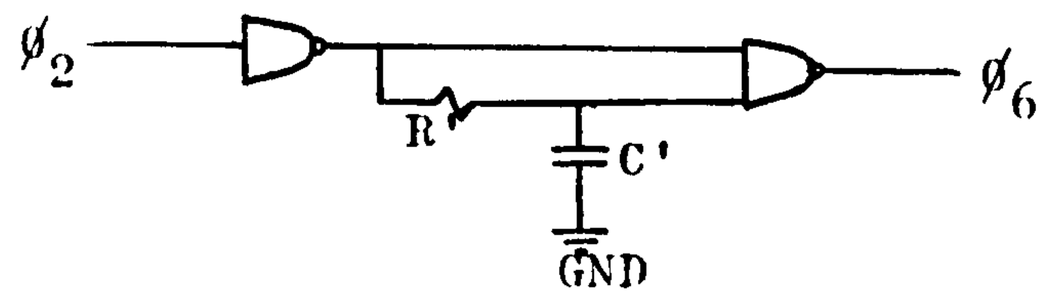


FIG: 7.2.7 GENERATION OF CONTROL SIGNAL $\overline{\phi}_6$

In fact, ϕ_6 and $\overline{\phi}_6$ can also be obtained from ϕ_2 and $\overline{\phi}_2$ with the circuits shown below.



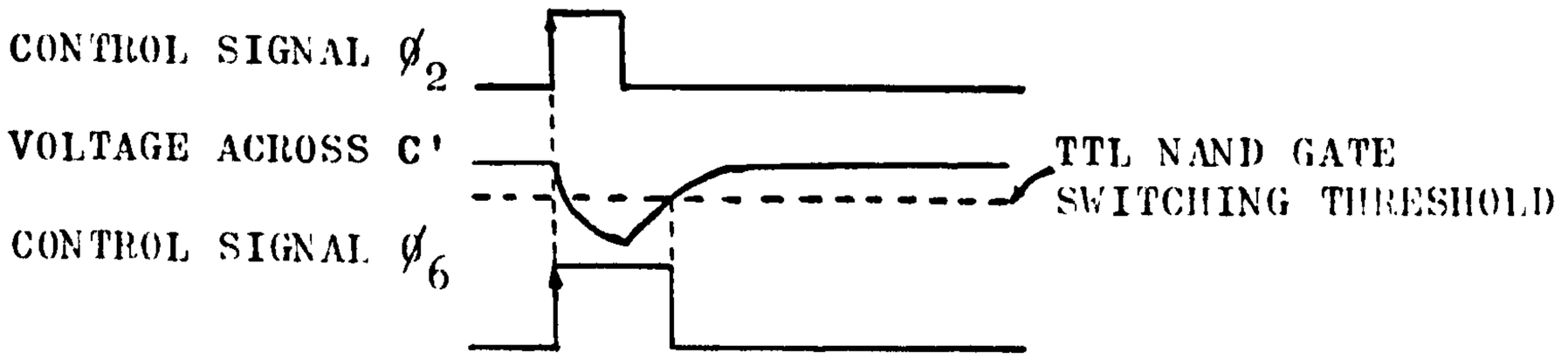


FIG: 7.2.8 GENERATION OF ϕ_6 FROM ϕ_2

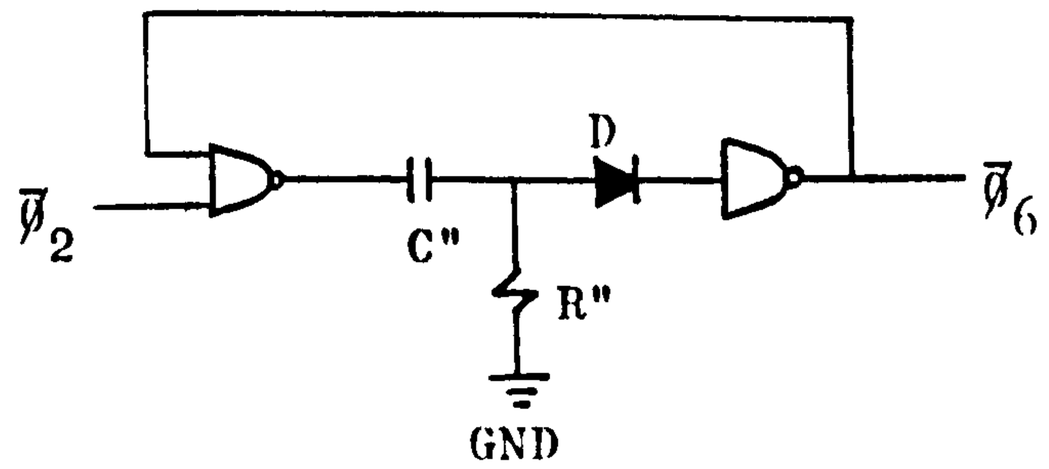


FIG: 7.2.9 GENERATION OF $\overline{\phi}_6$ FROM $\overline{\phi}_2$

The circuit shown in figure 7.2.8 is thus seen to have a pulse-stretching effect on ϕ_2 thereby producing ϕ_6. The time constant R'C' is chosen to be less than the width of the ϕ_2 pulse. The circuit shown in figure 7.2.9 represents a "differentiating" approach as opposed to the "integrating" approach used so far in producing control signals. The time constant R"C" should be chosen to be greater than the width of the $\overline{\phi}_2$ pulse and the waveforms of the circuit are shown below. The diode D is used as a blocking diode.

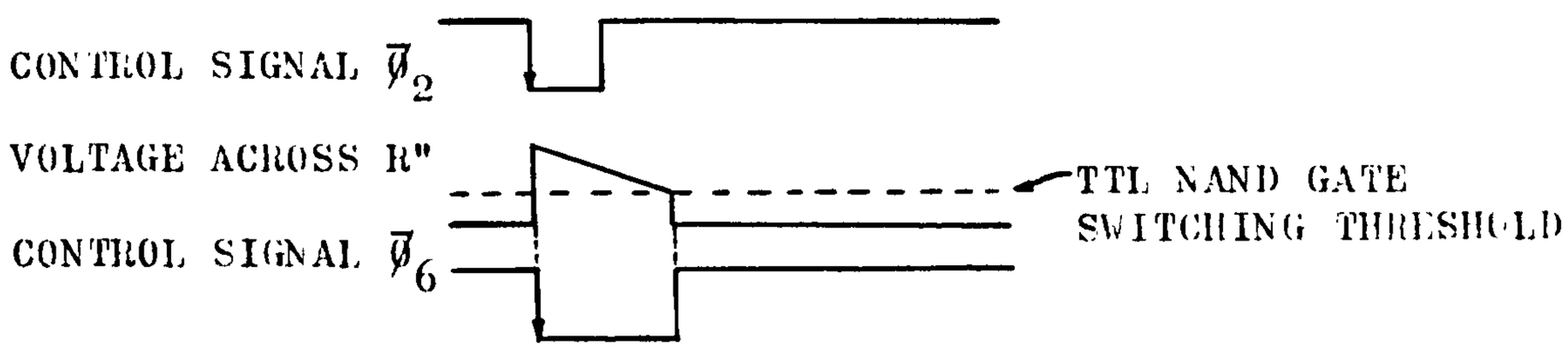


FIG: 7.2.10 WAVEFORMS OF FIGURE 7.2.9.

On the other hand, when the time constant R"C" is much less than the width of an applied input pulse, and if we further let the input pulse to be the status signal s_1, then the above circuit shown in figure 7.2.9 can also be used to generate the control signal ϕ_2 instead of the circuit shown in figure 7.2.1 previously.

Having dealt with the control and timing signals of the demonstration processor, we now include two methods used in the construction of the above processor to avoid the problem of "clock skew". Clock skew is a problem inherent to any synchronous digital system using high-speed TTL logic devices (e.g. Schottky TTL logic devices). It is independent of the system clock rate and affected only by the device speed, that is, its propagation delay. Ideally, all clock inputs of the devices in the demonstration processor should be activated simultaneously. In a practical system layout, this is not always possible. There are time differences, called clock skew, due to line propagation time and/or delay variations between different clock buffers. However, there are two ways to avoid clock skew problems. One is to minimize clock skew by driving all clock inputs in parallel from one source. That is, serial driving of clock inputs should be avoided as far as possible (e.g. obtaining a

clock pulse to drive one device from the clock input of a previous device in a chain of devices should be avoided). When a certain amount of serial clock driving is unavoidable, the second method of avoiding clock skew is to arrange clock delays carefully so that the clock delays run in the opposite direction of the data delays. This insures that the last device in the data chain is clocked first, and all clock skew problems are consequently eliminated. Furthermore, standard procedures of power supply decoupling, earthing and shielding of devices in the demonstration processor have been assumed in order to minimize noise and to ensure correct operations of all devices.

We now consider the problem of the synchronization of the clock signal of the demonstration processor with its control and timing signals. This is particularly important when the demonstration processor is required to work at its maximum possible rate. Consider a technique depicted below which is accurate to around 1nsec in synchronizing the system clock and its control and timing signals.

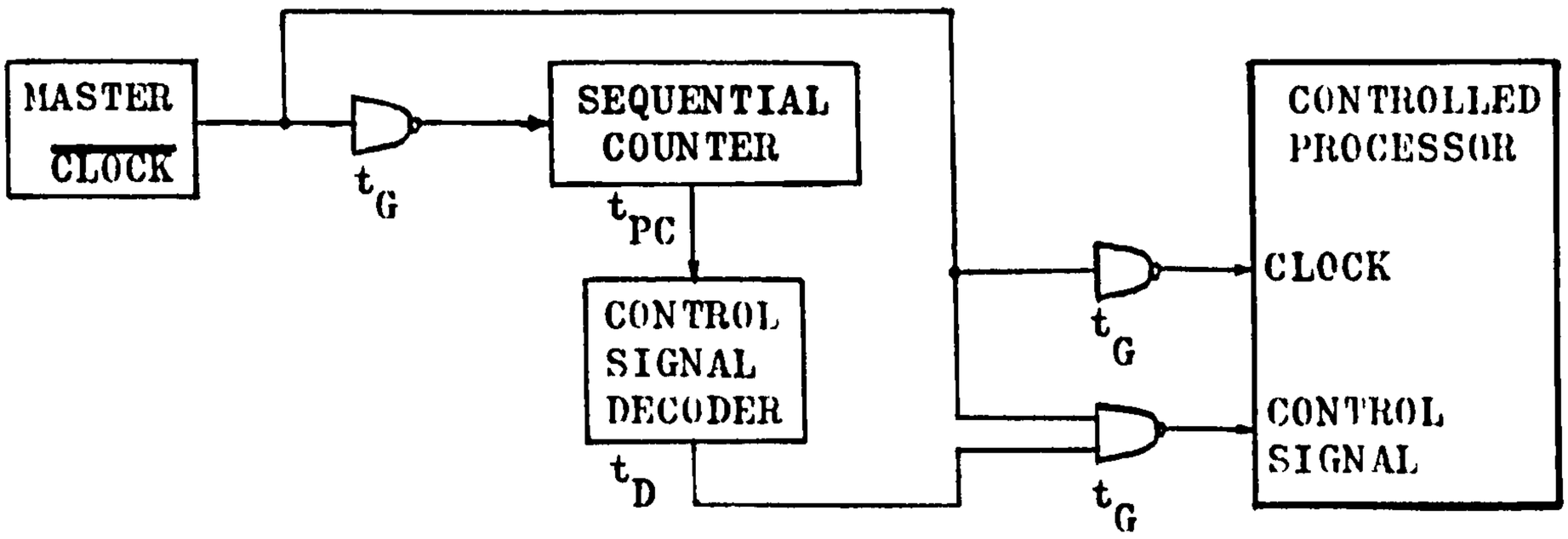


FIG: 7.2.11 BLOCK DIAGRAM OF SYSTEM LAYOUT FOR HIGH PRECISION SIGNAL SYNCHRONIZATION AND PERTINENT WAVEFORMS

The propagation delay of the program counter of the processor is represented by t_{PC} which is of the order of a few nanoseconds even when using Schottky TTL technology. The propagation delay in obtaining the status and control signals via some decoding logic is denoted by t_D and is of the order of a few nanoseconds. In an ordinary system layout, all these delays add up so that a control signal is obtained after a certain delay after the application of the clock signal. The system layout in figure 7.2.11 shows the synchronizing effect of the NAND gates which are used to minimize the above mentioned delay in obtaining a control signal to around 1nsec. The control signal is in the high (logical 1) state until the decoder output goes high. After the decoder output has gone high, the control signal remains high until $\overline{Ck}$ also goes high. After $\overline{Ck}$ has gone high, the inputs of the two NAND gates feeding the controlled system are the same. Therefore, the control signal and the clock signal are alike; their falling edges and the subsequent rising edges virtually coincide at the positive-going or rising edge of the next clock pulse. The only skew remaining is the difference between the propagation delays of the two NAND gates. With Schottky TTL, the difference between the gate-delays of the above two NAND gates is typically less than 1nsec.

In this section, we have so far discussed the control and timing circuitry of the serial-mode demonstration processor, and in the latter part of this chapter we shall see how they will be applied. At this juncture, it may be worth pointing out that the control and timing circuitry required in a parallel implementation of the proposed CBFCS algorithm described in chapter five are comparatively less sophisticated. However, as pointed out before, a parallel-mode implementation of the proposed CBFCS algorithm requires, on the other hand, much more hardware. The design of the control and timing circuitry of the demonstration processor built is oriented for high-speed operations. In particular, we have taken into account of the high-precision clock and control signals synchronization which is very important in high-speed operations to avoid undesirable propagation delays of devices. Particular attention has been paid to the clock skew problems inherent to the use of high-speed Schotiky TTL devices in additional to the initialization problem of the processor.

7.3 THE INPUT PORT (IP):

The input port (IP) consists of the shift registers SR_{x_n} and $SR_{x_{n-1}}$ shown below:

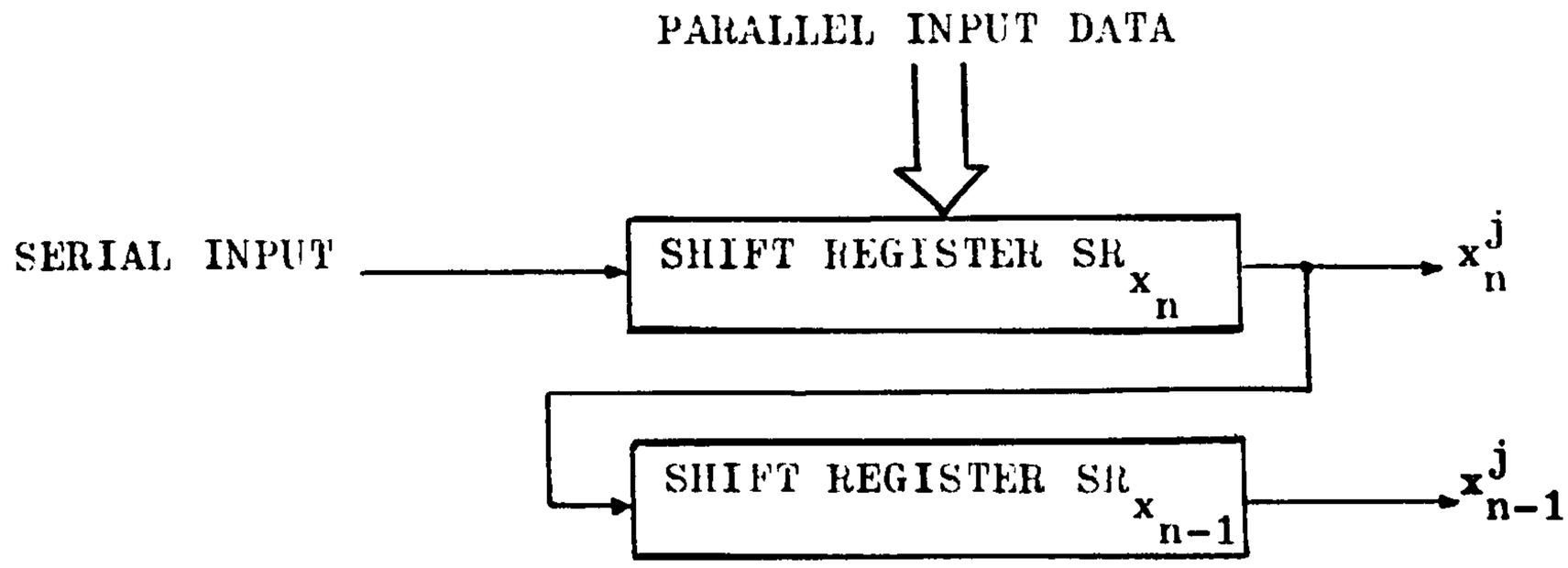


FIG: 7.3.1 STRUCTURE OF THE INPUT PORT (IP)

The outputs of the input port (IP) are the sequences $\{x_n^j\}$ and $\{x_{n-1}^j\}$ as shown above for addressing the CBFCS memory. It is implemented in hardware with two 9-bit TTL shift registers because of the 9-clock-cycle operation of the "psuedo-pipeline" demonstration processor. For serial data input, the least significant bit of the data is first shifted in, while the most significant end of the data is padded with an extra bit with a logical value same as the sign bit of the input data. For instance, when the sign bit has a logical value of 1, the extra bit is then a logical 1. For parallel data input, the 8-bit data word is loaded in parallel into the last eight bits of the shift register SR_{x_n} at the start of the first clock cycle, while the data bits are then serially shifted out in the remaining eight clock cycles of the computation of y_n.

In the demonstration processor, two packages of SN74198 (TTL 8-bit shift register with parallel/serial loading and inhibit mode control) and one package of SN74S74 were used as shown in figure 7.3.2. (For higher speeds of operation, shift registers may be formed by a slight modification of SN74S374, SN74S174 or SN74S175). To ensure proper operation, these shift registers are first cleared by the initialization pulse I at the start of the operation of the demonstration processor, i.e. when the power is first switched on.

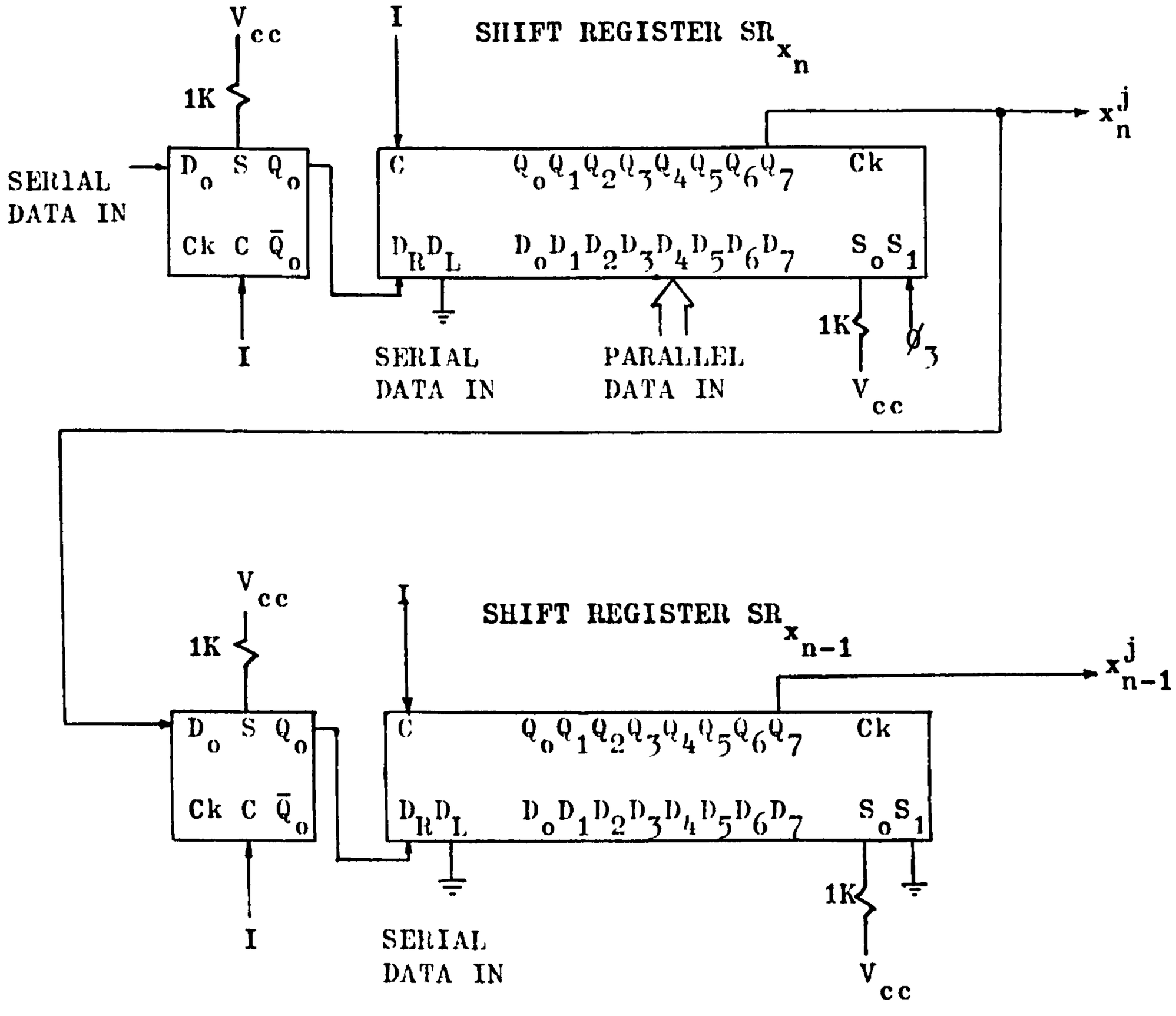


FIG: 7.3.2 LOGIC DIAGRAM OF THE INPUT PORT (IP)

When the demonstration processor is multiplexed to implement filters designed by the proposed design technique discussed in chapter three, the connections of the shift registers:

$$(SR_{x_n}, SR_{x_{n-1}})$$

shown above will be then wired as shown in figure 5.4.3 in chapter five. (For serial data input, the control input S_1 of the shift register SR_{x_n} will be taken low, that is, grounded). For instance, if an 8th order bandpass filter is to be implemented, data within the above shift register SR_{x_n} will then be made to recirculate

four times before a fresh input sample is to be clocked in. In the configuration shown in figure 5.4.3, only the shift register SR_{x_n} is connected for data recirculation, while the shift register $SR_{x_{n-1}}$ will have to be 36 bits long for the above 8th order filter as four second-order sections are required to make up an 8th order filter. On the other hand, if data in the shift register $SR_{x_{n-1}}$ are also made to recirculate four times as well, by taking its output back to its input as shown below:

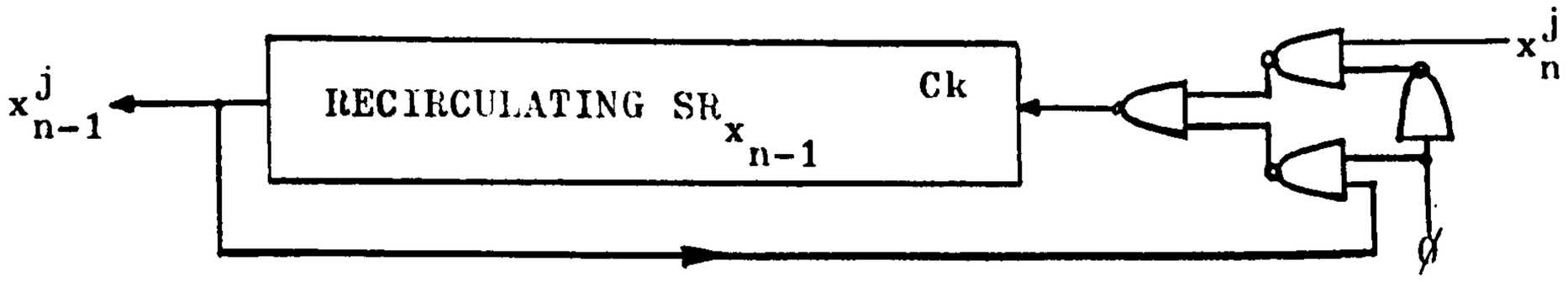


FIG: 7.3.3 RECIRCULATING $SR_{x_{n-1}}$

then the shift register $SR_{x_{n-1}}$ will only be 9 bits long. In figure 7.3.3, when the control ϕ is a logical 1 then the data in $SR_{x_{n-1}}$ will recirculate, while data x^j_n will be shifted in when ϕ is a logical 0. In order to maintain the proper delay required between the x_n and x_{n-1} samples, the serial data input of the now made recirculating shift register $SR_{x_{n-1}}$ is inhibited by taking ϕ high when the data in SR_{x_n} are being recirculated for the first three times. At the start of the fourth recirculating cycle of SR_{x_n} data are then serially shifted in $SR_{x_{n-1}}$ by taking ϕ low, and when the whole input sample x_n is shifted into $SR_{x_{n-1}}$, it is then made to recirculate four times. This operation now repeats for every fresh data sample to the multiplexed demonstration processor via the modified input port (IP) described above.

The above control signal ϕ which controls the operation of the above recirculating shift registers can be easily obtained by the divide-by-four synchronous gateless counter shown in figure 7.3.4 whose state table and diagram are depicted in figure 7.3.5. The output of this divide-by-four synchronous counter is decoded by a NAND gate to give ϕ while its clock input is fed by $\overline{\phi}_4$ from the main control and timing unit (CTU).

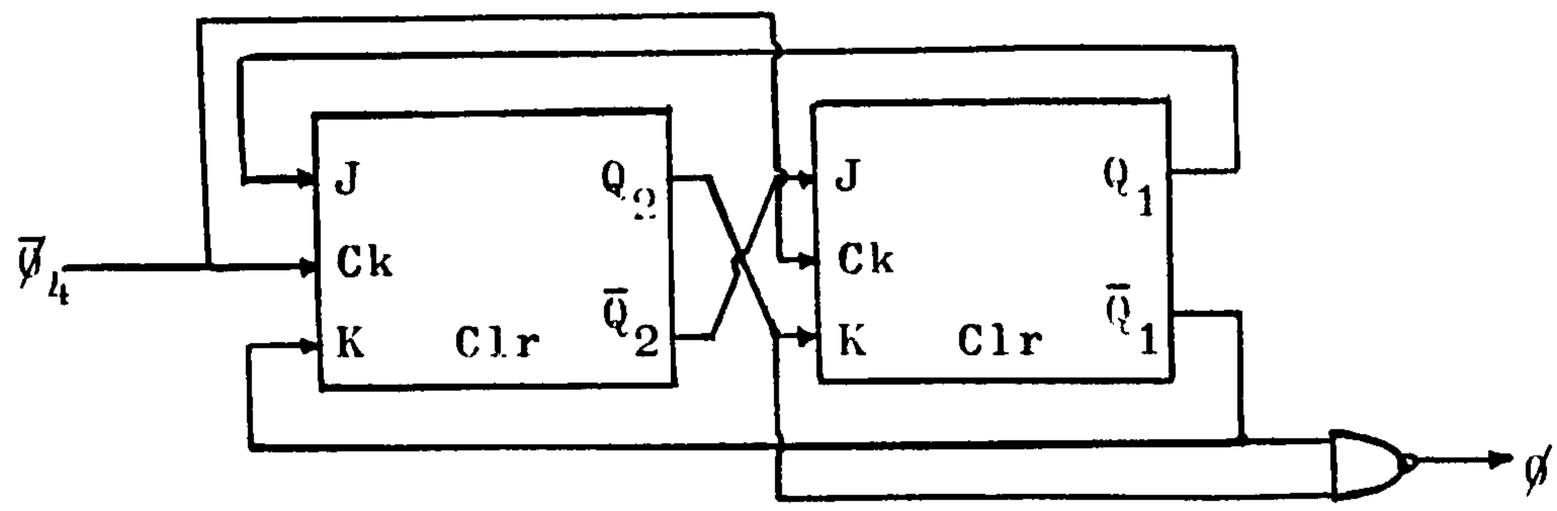
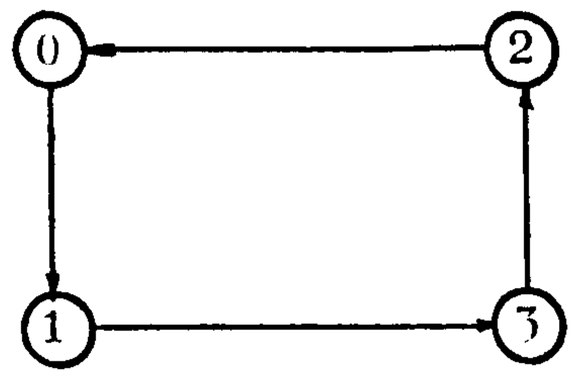


FIG: 7.3.4 GENERATION OF CONTROL SIGNAL ϕ

Q_2	Q_1	COUNT
0	0	1
0	1	2
1	1	3
1	0	4
0	0	5

REPEAT

STATE TABLE



STATE DIAGRAM

FIG: 7.3.5 STATE TABLE AND DIAGRAM OF A SYNCHRONOUS DIVIDE-BY-4 COUNTER

7.4 THE CBFCS MEMORY:

The precomputed "compensated partial products" of the proposed CBFCS algorithm are stored in the CBFCS memory. For a second-order section, the CBFCS memory structure is as shown in figure 7.4.1. The 32x16 random access memory bank is being made up of eight 16x4 bipolar Schottky TTL RAM's (SN74S189) which were chosen for the demonstration processor because they were the fastest TTL RAM's available then, having an access time of typically 25nsec. The use of RAM's in implementing the CBFCS memory makes possible a variable filter by selectively changing the contents of the RAM's. The precomputed "compensated partial products" are shown as "data-in" to the CBFCS memory in figure 7.4.1. These "compensated partial products" are written into the CBFCS memory via a bank of manual switches, repeatedly set to represent the individual values of these required "compensated partial products".

The Schottky TTL RAM's (SN74S189) used are tri-state RAM's so that their outputs can be wired-OR together. Unlike other TTL RAM's with open-collector outputs which requrie pull-up resistors tied to the power rail, these tri-state Schottky TTL RAM's have extra drive capability and can drive directly bus-organized/high capacitive load. The detail connections of the CBFCS memory are shown as a bus-organized system in figure 7.4.2.

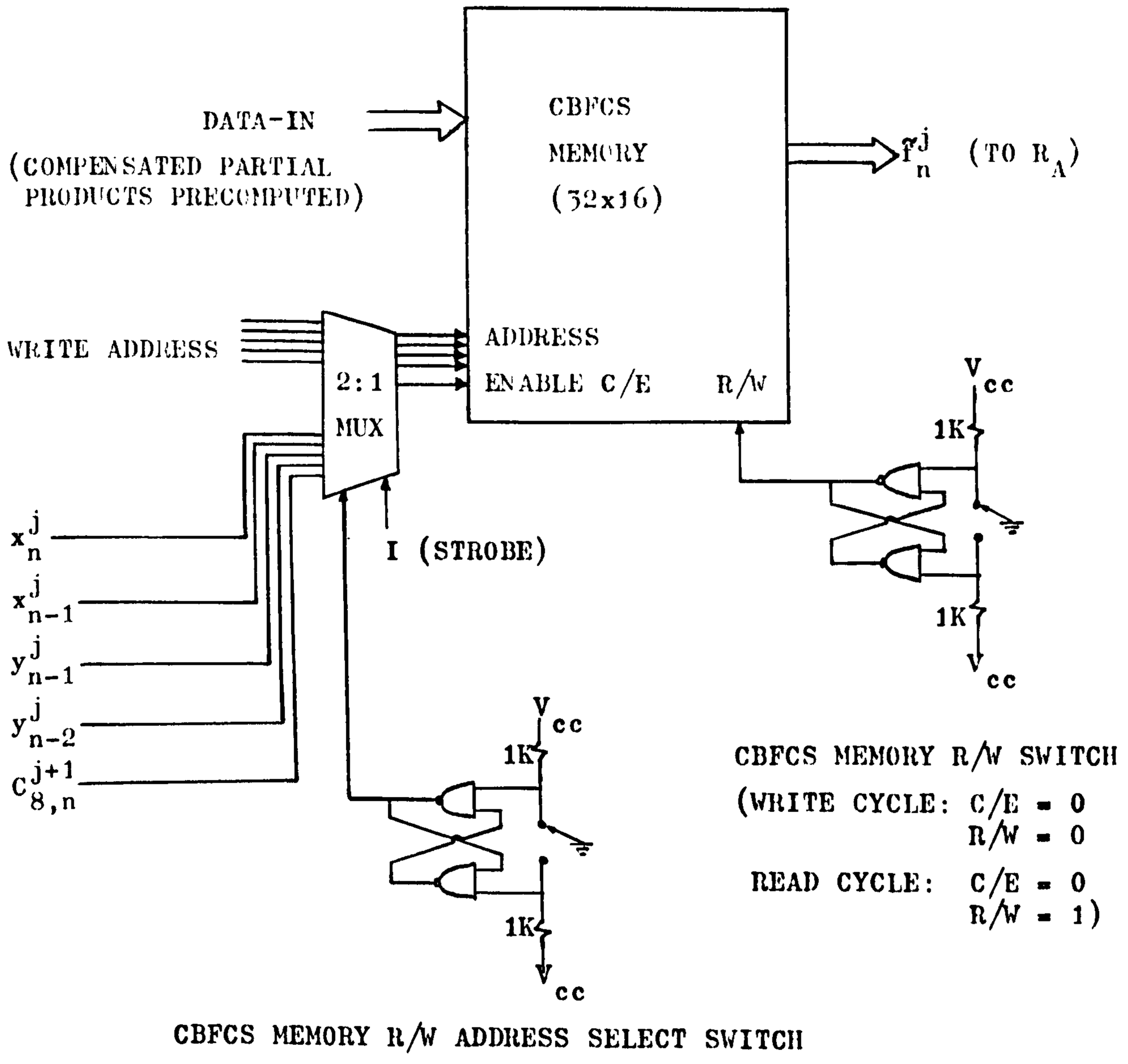


FIG: 7.4.1 BLOCK DIAGRAM SHOWING THE CBFCS MEMORY STRUCTURE FOR A SECOND-ORDER SECTION

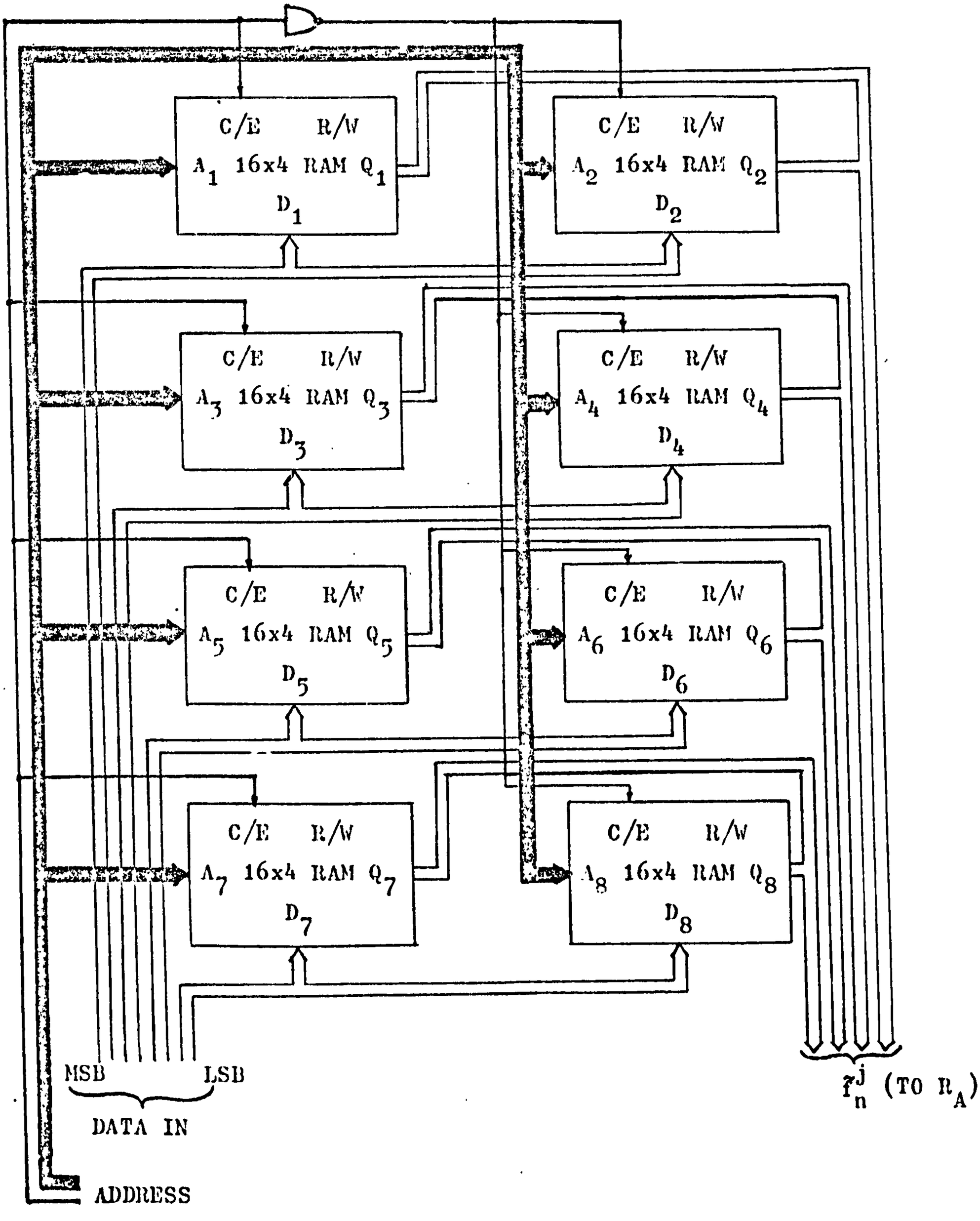


FIG: 7.4.2 BUS-ORGANIZED CBFCS MEMORY FOR A SECOND-ORDER SECTION

The "WRITE ADDRESS" and the "READ ADDRESS" are coupled to the address input of the CBFCS memory via the 2:1 multiplexer which was implemented by $1\frac{1}{4}$ packages of SN74S157 Schottky TTL multiplexer shown below:

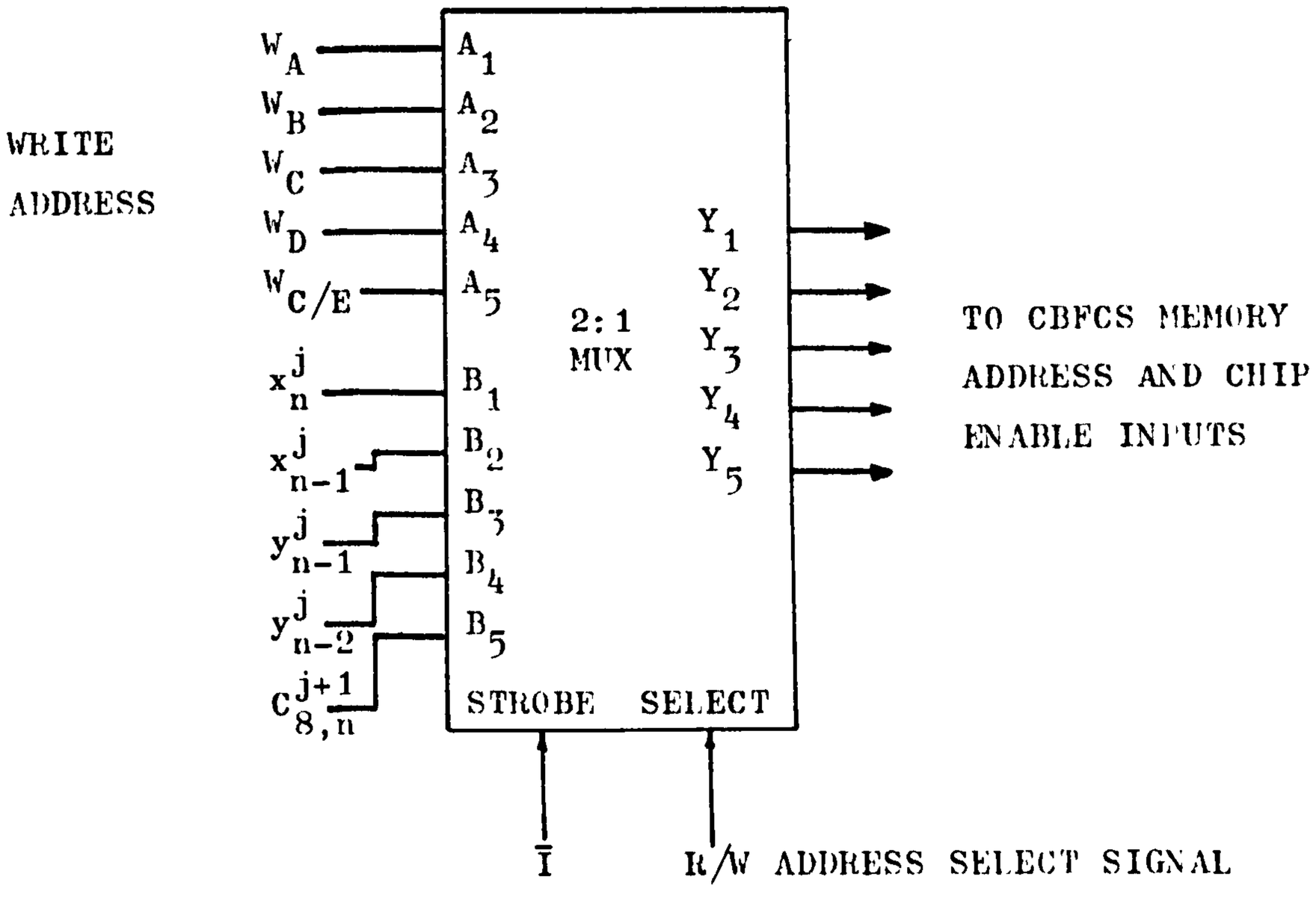


FIG: 7.4.3 THE CBFCS MEMORY INPUT MULTIPLEXER FOR SELECTING THE READ/WRITE ADDRESSES

When writing the precomputed "compensated partial products" into the CBFCS memory, a set of manual switches were used to provide the necessary "WRITE ADDRESS" for accessing the 32 locations of the CBFCS memory. When the above "compensated partial products" have been written into the CBFCS RAM's, then the demonstration processor is ready for filtering and the content of the CBFCS RAM's are read out by addressing the CBFCS memory via the "READ ADDRESS"

$$(x^j_n,\ x^j_{n-1},\ y^j_{n-1},\ y^j_{n-2},\ C^{j+1}_{8,n})$$

When the demonstration processor is multiplexed to implement, say, an 8th order bandpass filter designed by the proposed design technique discussed in chapter three, an extra 4:1 memory address multiplexer has to be used and connected as shown in figure 7.4.4 and described below.

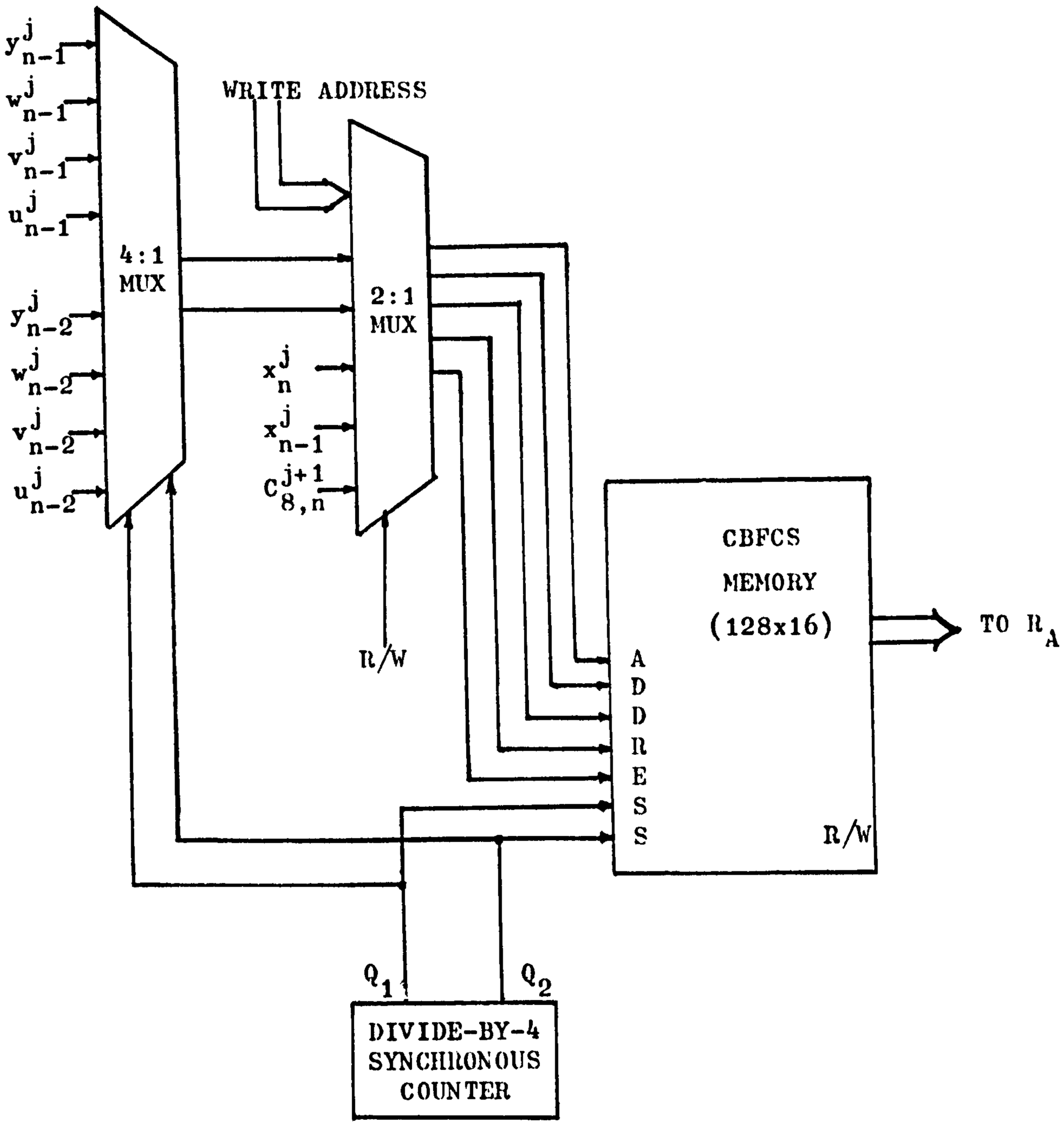


FIG: 7.4.4 BLOCK DIAGRAM SHOWING THE (128x16) CBFCS MEMORY STRUCTURE FOR A MULTIPLEXED EIGTH-ORDER FILTER

As described in section 5.4, the CBFCS memory required in the multiplexed 8th order bandpass filter is now four times that shown in figure 7.4.2, organized basically as four seperate sections in parallel to store the pre-computed "compensated partial products" of the four parallel second-order sections (elemental filters) that make up the above 8th order filter. These four memory sections that make up the 128x16 CBFCS memory are sequentially selected, as shown in figure 7.4.4, by two select signals which are the outputs (Q_1, Q_2) of a divide-by-4 synchronous counter whose logic diagram will be shown in figure 7.13.3. The 4:1 multiplexer (SN74S153) shown in figure 7.4.4, enables the outputs of the output ports (whose structures are identical as will be discussed in section 7.13) of the above four parallel second-order sections of the multiplexed 8th order filter to address the 128x16 CBFCS memory sequentially in the multiplexed operation of the demonstration processor as discussed in section 5.4. The select inputs of the 4:1 multiplexer are also fed by the (Q_1, Q_2) outputs of the above divide-by-4 synchronous counter. Since the CBFCS memory for the multiplexed demonstration processor is implemented with RAM's, a 2:1 multiplexer, in cascade with the 4:1 multiplexer, is also required to couple the "READ ADDRESS" and the "WRITE ADDRESS" to the address input of the 128x16 CBFCS memory as shown in figure 7.4.4

7.5 THE REGISTER R_A:

The outputs of the CBFCS memory are connected to a 16-bit parallel-in-parallel-out register which is the register R_A shown in figure 7.1.1. As mentioned previously, the introduction of R_A has made the demonstration processor into a "psuedo-pipeline" system and enabled concurrent operations of the accessing of the CBFCS memory and the "byte-sliced" summation of the "compensated partial products" of f_n as described in detail in section 5.3.2 in chapter five. As shown in figure 7.1.1, R_A stores each output $\tilde{f}_n^j$ of the CBFCS memory before loading it into the "Adderless-Multiplierless Unit" (AMU). R_A is first initialized by the initialization pulse I when power is switched on, and cleared at the start of every first clock cycle of the 9-clock-cycle operation of the demonstration processor as tabulated in table 7.5.1. The output of R_A is thus seen, from table 7.5.1, to be a delayed version of the output of the CBFCS memory.

CLOCK CYCLE	OUTPUT OF THE CBFCS MEMORY	CONTENT OF R_A
1	$\bar{f}_n^7$	0
2	$\bar{f}_n^6$	$\bar{f}_n^7$
3	$\bar{f}_n^5$	$\bar{f}_n^6$
4	$\bar{f}_n^4$	$\bar{f}_n^5$
5	$\bar{f}_n^3$	$\bar{f}_n^4$
6	$\bar{f}_n^2$	$\bar{f}_n^3$
7	$\bar{f}_n^1$	$\bar{f}_n^2$
8	$\bar{f}_n^o$	$\bar{f}_n^1$
9	$\bar{f}_n^o$	$\bar{f}_n^o$

TABLE 7.5.1 OPERATION OF REGISTER R_A

R_A was implemented with two packages of SN74198 as shown in figure 7.5.1 below. (For higher speeds of operation, the register R_A can be implemented with SN74S374, SN74S174 or SN74S175 instead.)

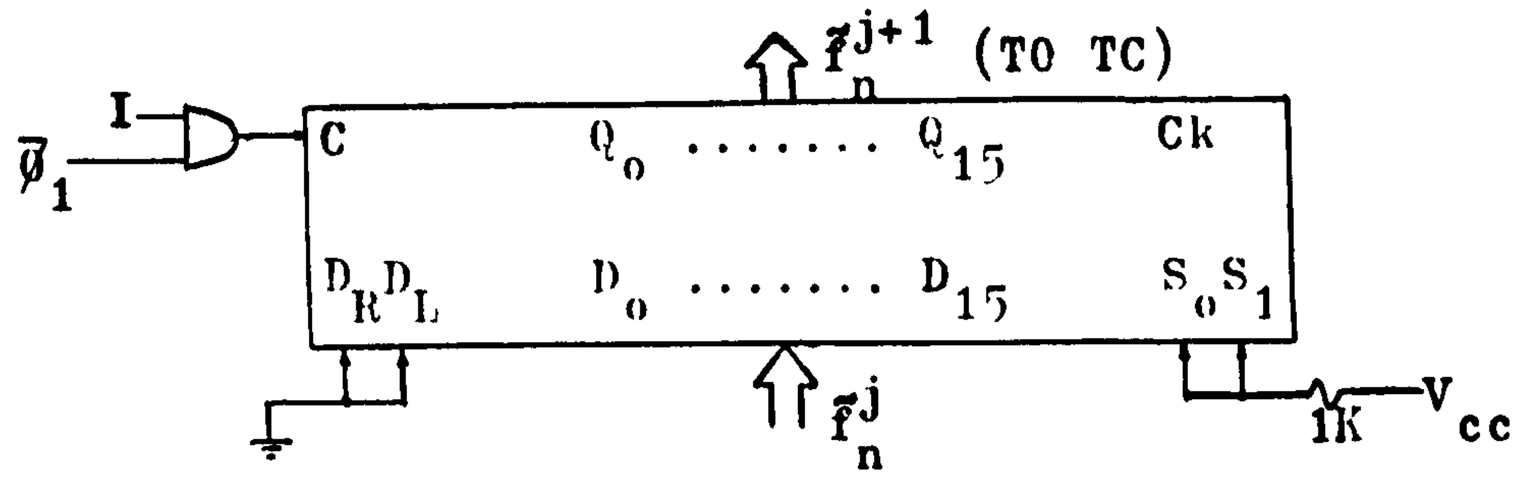


FIG: 7.5.1 LOGIC DIAGRAM OF REGISTER R_A

The output of register R_A is connected to the 16-bit "True-Complementer" (TC) as shown above. The control to the clear input

C of the SN74198 packages is $\overline{\emptyset}_1$.I as shown. The above 16-bit "True-Complementer" is discussed in the next section.

7.6 THE TRUE-COMPLEMENTER (TC):

From equation (5.2.11), we have, for the demonstration processor,

$$f_n = \sum_{j=1}^{7} 2^{-j}.f(x_n^j,\ x_{n-1}^j,\ y_{n-1}^j,\ y_{n-2}^j,\ C_{8,n}^{j+1}) - f(x_n^o,\ x_{n-1}^o,\ y_{n-1}^o,\ y_{n-2}^o,\ C_{8,n}^1) \qquad (7.6.1)$$

Therefore, a two's complement "byte-sliced" subtraction is required in the last clock cycle of the computation of f_n as given in the above equation. For this purpose, a 16-bit "True-complementer" is used to one's-complement the output of the register R_A at the last clock cycle, while a forced carry is input to the least significant carry input of the "Adderless-Multiplierless Unit" (AMU) to effect the subtraction in equation (7.6.1) above. The generation and connection of the above forced carry at the last clock cycle will be discussed in the next section, while figure 7.6.1 below shows the logic diagram of the "True-Complementer" (TC).

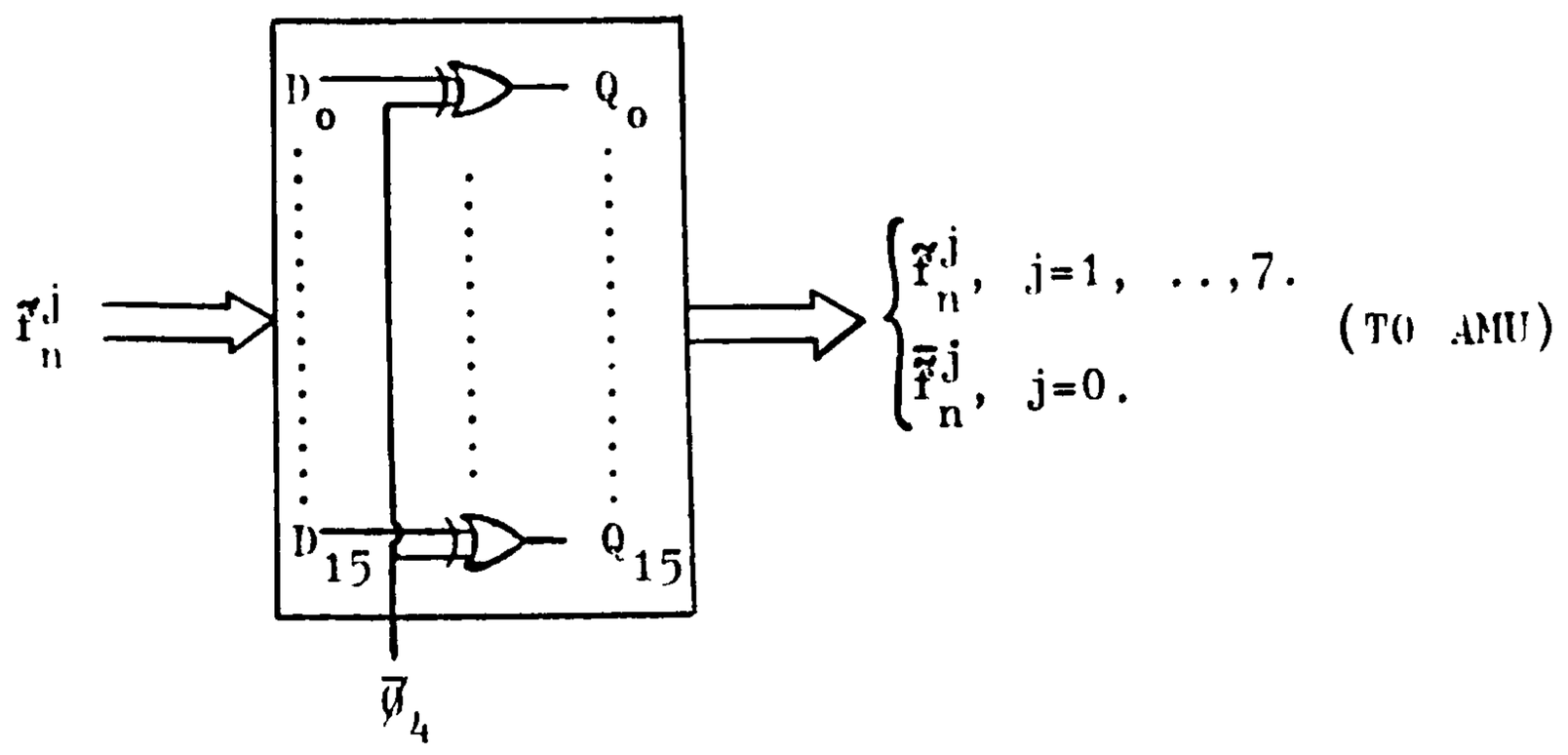


FIG: 7.6.1 16-BIT "TRUE-COMPLEMENTER" (TC)

The above "True-Complementer" was made up of 16 EX-OR gates (i.e. four packages of SN74S86) in the demonstration processor, and controlled by the control signal $\overline{\emptyset}_4$ as shown above. The output of

the above "True-Complementer" (TC) is hard-wired to one of the two summation inputs of the "Adderless-Multiplierless Unit" (AMU), and f_n^j shown in figure 7.6.1 is the one's complement or inverted value of f_n^j. Hence, for j=1, ...,7 the output of the "True-Complementer will be f_n^j since the control signal $\overline{\emptyset}_4$=0. On the other hand, when j=0, the output of the "True-Complementer" (TC) will be f_n^j instead, since the control signal then takes on a logical 1, that is $\overline{\emptyset}_4$=1. The value of f_n^j when j=0 and a simultaneous forced carry mentioned above will form the term:

$$-f(x_n^o,\ x_{n-1}^o,\ y_{n-1}^o,\ y_{n-2}^o,\ C_{8,n}^1)$$

in equation (7.6.1) above to effect the required "byte-sliced" subtraction in the last cycle of the computation of f_n.

As discussed in the previous section, for higher speeds of operation, the register R_A shown in figure 7.5.1 will be implemented with registers, say SN74S175, such that the inverted value (i.e. the one's complement value) of its output is also available on chip. In this instance, a 16-bit 2:1 multiplexer (made up of four packages of SN74S158) will be used instead of the above 16-bit "True-Complementer" (TC) to route f_n^j or f_n^j to the "Adderless-Multiplierless Unit" (AMU) in the respective clock cycles as described previously. We shall next discuss the structure of the "Adderless-Multiplierless Unit" (AMU) of the demonstration processor in the following section.

7.7 THE"ADDERLESS-MULTIPLIERLESS UNIT" (AMU):

As mentioned in chapter five, the "Adderless-Multiplierless Unit" (AMU) of the proposed CBFCS approach is implemented with standard TTL 4-bit full adders wired as special combinational logic units. In the demonstration processor, four such adders (4xSN74283) were used and wired as shown in figure 7.7.1. The 16-bit AMU as shown in figure 7.7.1 is multiplierless, and is also adderless as the 4-bit TTL full adders are not connected to operate as a 16-bit conventional full adder. Since the carry propagation path is being "sectioned" at the point C_8 only, the four 4-bit adders are seperated into two groups. Each of the two groups has thus ripple-carry propagation between adders so that the typical summation time within

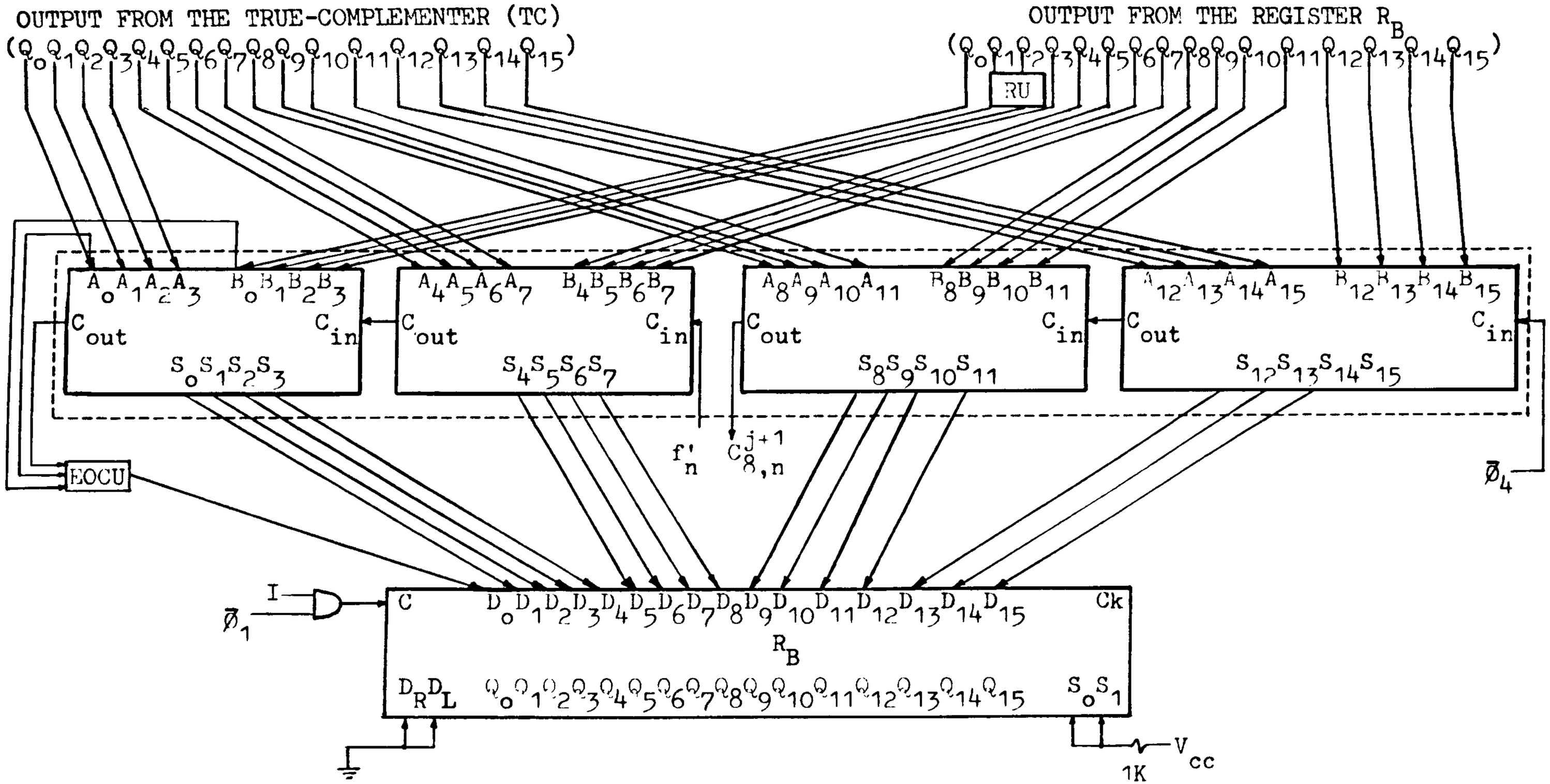


FIG: 7.7.1 LOGIC DIAGRAM SHOWING THE STRUCTURE OF THE 16-BIT "ADDERLESS-MULTIPLIERLESS UNIT" (AMU) OF THE DEMONSTRATION PROCESSOR AS THAT ENCLOSED IN THE ABOVE DOTTED LINES

each group is 23nsec. Furthermore, in the "byte-sliced" summation operation of the 16-bit AMU, the 16-bit inputs to the AMU are each "byte-sliced" into two 8-bit bytes. These 8-bit bytes are handled by the above two groups of adders simultaneously so that the typical summation time of the AMU is thus 23nsec. Had the above four TTL adders been wired as a conventional 16-bit full adder, the summation time would have been typically 43 nsec instead. Hence, the AMU of the demonstration processor has enabled a two-fold increase in speed as compared to a conventional 16-bit full adder using the same four TTL standard 4-bit full adders, by sectioning the 16-bit worst case carry propagation path at the point C_8 so as to reduce it down to a maximum of only eight bits long.

As mentioned previously, the carry-out $C_{8,n}^{j+1}$ from the AMU is brought forward to address the CBFCS memory as shown in figure 7.4.1 and also to the compensation units CU' and CU" (see a later section). The output f_n' of the compensation unit CU' is fed back to the AMU as shown in figure 7.7.1 every 9th clock cycle of the computation of f_n. The control signal $\overline{\emptyset}_4$, acting as a forced carry, is wired to the carry input at the least significant end of the AMU as shown in figure 7.7.1 to effect the subtraction in equation (7.6.1) in conjunction with the "True-Complementer" (TC) as described in the previous section, in the 9th clock cycle of the computation of f_n. Furthermore, the carry out of the most significant end of the AMU is fed to the "End-Overflow Correction Unit" (EOCU) as shown in figure 7.7.1 and also to the "Overflow Detection And Saturation Arithmetic" (ODSAU) as will be described in a later section.

The 16-bit output of the AMU is hard-wired to the input of the register R_B with one bit right-shift as shown in figure 7.7.1. The 16-bit output of the register R_B, part of which also passes through the rounding unit RU, is connected to the other summation input of the AMU as shown in figure 7.7.1 to complete the feedback loop of accumulation in the "byte-sliced" summation of the compensated partial products of f_n. Part of the 16-bit output of the AMU is loaded in parallel into the shift register SR_{y^*} at the start of every 1st clock cycle in the computation of f_n^{n-1} as will be described in a later section. In the demonstration processor, the output is represented to 8-bit accuracy including sign, hence, only eight bits

of the 16-bit output of the AMU at the end of the computation of f_n are significant and being stored for further processing to yield y_{n-1} as previously pointed out in section 7.1. Had the filter variables of the demonstration processor been represented by 16-bit two's complement numbers, then the full 16-bit output of the above AMU would have been utilised to yield a 16-bit y_{n-1}.

7.8 THE REGISTER R_B:

The 16-bit output of the AMU is wired to the input of the register R_B as shown in figure 7.7.1 and the output of the register R_B is fed back to one of the two summation inputs of the AMU as previously described. Hence, R_B forms an accumulator in conjunction with the AMU in the demonstration processor for the "byte-sliced" summation of the compensated partial products of f_n. Since R_B is a 16-bit parallel-in-parallel-out register, two packages of SN74198 were used to implement it as shown in figure 7.7.1. (For higher speeds of operation, the register R_B can be implemented with SN74S374, SN74S174, or SN74S175 instead). The control signal to the clear input C of the SN74198 packages is $\overline{\emptyset}_1$.I as shown, such that the register R_B is first initialized by the initialization pulse I when power is first switched on, and cleared at the start of every first clock cycle of the 9-clock-cycle operation of the demonstration processor.

7.9 THE "END-OVERFLOW CORRECTION UNIT" (EOCU):

Sometimes, in the summation of the compensated partial products of f_n, there may be "end-overflows" which result when the "depleted sums" (at the output of the AMU) are out-of-range. Furthermore, these "depleted sums" have to be right-shifted by one bit, and as pointed out in section 5.3.3, when right-shifting two's complement numbers, modifications have to be made in order to restore the proper weightings of the numbers being shifted, a special dual-purpose "End-Overflow Correction Unit" (EOCU) was designed and incorporated in the demonstration processor to ensure correct operations of the AMU.

There are a number of ways to solve the above problem; one of which is to examine the sign bit of a "depleted sum" at the output of the AMU at any time, and the sign bits of the numbers summed. This

can best be shown by way of examples. As we are examining a total of three bits, there are altogether $2^3=8$ cases of interest. For the purpose of examining these eight cases, we shall denote the carry out of the most significant end of the AMU shown in figure 7.7.1 as $C^j_{o,n}$ for the present clock cycle, with the two numbers to be summed, at the two summation inputs of the AMU as:

$$A^j_n = (A^j_{o,n},\ A^j_{1,n},\ \ldots\ldots,\ A^j_{15,n})$$

from the output of the 16-bit "True-Complementer" (TC), and

$$B^j_n = (B^j_{o,n},\ B^j_{1,n},\ \ldots\ldots,\ B^j_{15,n})$$

from the output of the 16-bit register R_B. (Although, as shown in figure 7.7.1, bits 1 and 2 of the output of R_B pass through the rounding unit RU, the output of R_B can be regarded as an input to the AMU since the sole purpose of the rounding unit RU is to perform pre-rounding at the start of the 1st clock cycle when the content of R_B is cleared). Also the "depleted sum" at the 16-bit output of the AMU corresponding to the above two numbers is denoted as:

$$S^j_n = (S^j_{o,n},\ S^j_{1,n},\ \ldots\ldots,\ S^j_{15,n})$$

(The various outputs of the AMU during the 9-clock-cycle operation of the demonstration processor has already been shown in table 7.1.1). In the next clock cycle, the carry out of the most significant end of the AMU will be $C^{j-1}_{o,n}$ and the inputs and output of the AMU become:

$$A^{j-1}_n = (A^{j-1}_{o,n},\ A^{j-1}_{1,n},\ \ldots\ldots,\ A^{j-1}_{15,n})$$

$$B^{j-1}_n = (B^{j-1}_{o,n},\ B^{j-1}_{1,n},\ \ldots\ldots,\ B^{j-1}_{15,n})$$

$$S^{j-1}_n = (S^{j-1}_{o,n},\ S^{j-1}_{1,n},\ \ldots\ldots,\ S^{j-1}_{15,n})$$

We now proceed to consider the above mentioned eight possible cases of interest in designing the "End-Overflow Correction Unit" as below.

Case 1: $S^j_{o,n}=0$, $A^j_{o,n}=0$ and $B^j_{o,n}=0$

$$\begin{array}{rl} & 0.0100000000000000 \longleftarrow A^j_n \\ & \underline{+0.0100000000000000} \longleftarrow B^j_n \\ C^j_{o,n} \longrightarrow (0) & 0.1000000000000000 \longleftarrow S^j_n \end{array}$$

In this case, $A^j_n=0.25$, $B^j_n=0.25$ and $S^j_n=0.5$ so that no end-overflow has occurred. As shown in figure 7.7.1, the output of the "End-Overflow Correction Unit" (EOCU) and the output of the AMU with one bit right-shift are hard-wired to the input of the register R_B. The output of the "End-Overflow Correction Unit" (EOCU) is now defined as the "End-Overflow Correction Function ψ". To ensure correct operations of the AMU, the value of ψ is chosen in this case to be 0 such that in the next clock cycle, $B^{j-1}_n=0.0100000000000000$.

Case 2: $S^j_{o,n}=0$, $A^j_{o,n}=1$ and $B^j_{o,n}=0$

$$\begin{array}{rl} & 1.1000000000000000 \longleftarrow A^j_n \\ & \underline{+0.1000000000000000} \longleftarrow B^j_n \\ C^j_{o,n} \longrightarrow (1) & 0.0000000000000000 \longleftarrow S^j_n \end{array}$$

In this case, $A^j_n=-0.5$, $B^j_n=0.5$ and so $S^j_n=0$ such that no end-overflow has occurred. The value of ψ is therefore chosen to be 0 in this case such that in the next clock cycle $B^{j-1}_n=0.0000000000000000$.

Case 3: $S^j_{o,n}=0$, $A^j_{o,n}=0$ and $B^j_{o,n}=1$

$$\begin{array}{rl} & 0.1000000000000000 \longleftarrow A^j_n \\ & \underline{+1.1000000000000000} \longleftarrow B^j_n \\ C^j_{o,n} \longrightarrow (1) & 0.0000000000000000 \longleftarrow S^j_n \end{array}$$

In this case, $A^j_n=0.5$, $B^j_n=-0.5$ and so $S^j_n=0$ as in the last case, such that no end-overflow has occurred and again the value of ψ is made 0.

Case 4: $S^j_{o,n}=0$, $A^j_{o,n}=1$ and $B^j_{o,n}=1$

$$\begin{array}{rl} & 1.1000000000000000 \longleftarrow A^j_n \\ & \underline{+1.0010000000000000} \longleftarrow B^j_n \\ C^j_{o,n} \longrightarrow (1) & 0.1010000000000000 \longleftarrow S^j_n \end{array}$$

In this case, $A_n^j=-0.5$, $B_n^j=-0.875$ so that their sum should be -1.375. However, instead of the correct sum, we have, in the above example, $S_n^j=0.625$ which is wrong both in sign and magnitude. Hence, an out-of-range end-overflow has occurred. We note that after one bit right-shift, the above sum -1.375 should become -0.6875 which is an in-range number. Hence, in the next clock cycle, we should have $B_n^{j-1}=1.0101000000000000$. The value of ψ is consequently chosen to be 1, to ensure correct operations of the AMU.

Case 5: $S_{o,n}^j=1$, $A_{o,n}^j=0$ and $B_{o,n}^j=0$

$$\begin{array}{rl} & 0.1000000000000000 \leftarrow A_n^j \\ & +0.1000000000000000 \leftarrow B_n^j \\ C_{o,n}^j \longrightarrow (0) & 1.0000000000000000 \leftarrow S_n^j \end{array}$$

In this case, $A_n^j=0.5$ and $B_n^j=0.5$ so that their sum should be 1. But, instead of the correct sum, we have, as shown above, $S_n^j=-1$ which has the wrong sign attached. We note that after one bit hard-wired right-shift, the correct sum should become 0.5 which is, as in the last case, an in-range number. Hence, in the next clock cycle, we should have $B_n^{j-1}=0.1000000000000000$. The value of ψ is thus chosen to be 0.

Case 6: $S_{o,n}^j=1$, $A_{o,n}^j=0$ and $B_{o,n}^j=1$

$$\begin{array}{rl} & 1.1010000000000000 \leftarrow A_n^j \\ & +0.0100000000000000 \leftarrow B_n^j \\ C_{o,n}^j \longrightarrow (0) & 1.1110000000000000 \leftarrow S_n^j \end{array}$$

In this case, $A_n^j=-0.375$, $B_n^j=0.25$ and so $S_n^j=-0.125$ such that no end-overflow has occurred. The value of ψ is therefore chosen to be 1 such that in the next clock cycle $B_n^{j-1}=1.1111000000000000$.

Case 7: $S_{o,n}^j=1$, $A_{o,n}^j=1$ and $B_{o,n}^j=0$

$$\begin{array}{rl} & 0.0100000000000000 \leftarrow A_n^j \\ & +1.1010000000000000 \leftarrow B_n^j \\ C_{o,n}^j \longrightarrow (0) & 1.1110000000000000 \leftarrow S_n^j \end{array}$$

In this case, A_n^j=0.25, B_n^j=-0.375 and so S_n^j=-0.125 as in the last case, such that no end-overflow has occurred and again the value of ψ is made 1.

Case 8: $S_{o,n}^j$=1, $A_{o,n}^j$=1 and $B_{o,n}^j$=1

$$
\begin{array}{rl}
 & 1.1000000000000000 \longleftarrow A_n^j \\
 & +1.1000000000000000 \longleftarrow B_n^j \\
C_{o,n}^j \longleftarrow & (1)1.0000000000000000 \longleftarrow S_n^j
\end{array}
$$

In this case, A_n^j=-0.5, B_n^j=-0.5 and so S_n^j=-1 which is in-range such that ψ is chosen to be 1 for B_n^{j-1}=1.1000000000000000 in the next clock cycle.

The above eight cases can now be summarized in the table below:

CASE	$A_{o,n}^j$	$B_{o,n}^j$	$C_{o,n}^j$	$S_{o,n}^j$	$B_{o,n}^{j-1}$	$B_{1,n}^{j-1}$	ψ
1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0
3	0	1	1	0	0	0	0
4	1	1	1	0	1	0	1
5	0	0	0	1	0	1	0
6	0	1	0	1	1	1	1
7	1	0	0	1	1	1	1
8	1	1	1	1	1	1	1

TABLE 7.9.1: STATE TABLE FOR THE END-OVERFLOW CORRECTION FUNCTION

In the above cases, it is noticed that end-overflows occur in cases 4 and 5 only where the End-Overflow Correction Function ψ takes on the value of $C_{o,n}^j$. In the other six cases where no end-overflow has occurred, the value of ψ is then made equal to $S_{o,n}^j$. Hence, in the mapping:

$$(\psi, S_{o,n}^j, S_{1,n}^j, \ldots., S_{14,n}^j) \Rightarrow (B_{o,n}^{j-1}, B_{1,n}^{j-1}, \ldots.., B_{15,n}^{j-1})$$

which represents the input-output characteristics of the register R_B,

we notice that:

$$(S^j_{o,n},\ S^j_{1,n},\ .....,\ S^j_{14,n}) = (B^{j-1}_{1,n},\ B^{j-1}_{2,n},\ .....,\ B^{j-1}_{15,n})$$

while $\psi = B^{j-1}_{o,n}$ is a function of the set $(A^j_{o,n},\ B^j_{o,n},\ C^j_{o,n})$ which can be found from the Karnaugh map below:

$C^j_{o,n}$ \ $A^j_{o,n}.B^j_{o,n}$	00	01	11	10
0	0	1	X	1
1	X	0	1	0

KARNAUGH MAP FOR FUNCTION ψ

Therefore, the function ψ can be written as:

$$\psi = \bar{A}^j_{o,n}.B^j_{o,n}.\bar{C}^j_{o,n} + A^j_{o,n}.B^j_{o,n}.C^j_{o,n} + A^j_{o,n}.\bar{B}^j_{o,n}.\bar{C}^j_{o,n}$$

$$= (\bar{A}^j_{o,n}.B^j_{o,n}.\bar{C}^j_{o,n})(A^j_{o,n}.B^j_{o,n}.C^j_{o,n})(A^j_{o,n}.\bar{B}^j_{o,n}.\bar{C}^j_{o,n})$$

which can be implemented with four 3-input NAND gates and four inverters. However, to save propagation delay and package count (and hence power consumption) we further manipulate the above expression of ψ by making the "don't care" condition (in the above Karnaugh map) for $\bar{A}^j_{o,n}.\bar{B}^j_{o,n}.C^j_{o,n}=1$, such that,

$$\psi = (\bar{A}^j_{o,n}.\bar{B}^j_{o,n} + A^j_{o,n}.B^j_{o,n})C^j_{o,n} + (\bar{A}^j_{o,n}.B^j_{o,n} + A^j_{o,n}.\bar{B}^j_{o,n})\bar{C}^j_{o,n}$$

$$= A^j_{o,n} \oplus B^j_{o,n} \oplus C^j_{o,n}$$

which can therefore be implemented by a 3-input EX-OR gate. However, two 2-input EX-OR gates (i.e. half a package of SN74886) were used owing to the fact that $C^j_{o,n}$, being a carry bit, would only appear after a finite amount of delay after the application of A^j_n and B^j_n to the two summation inputs of the AMU.

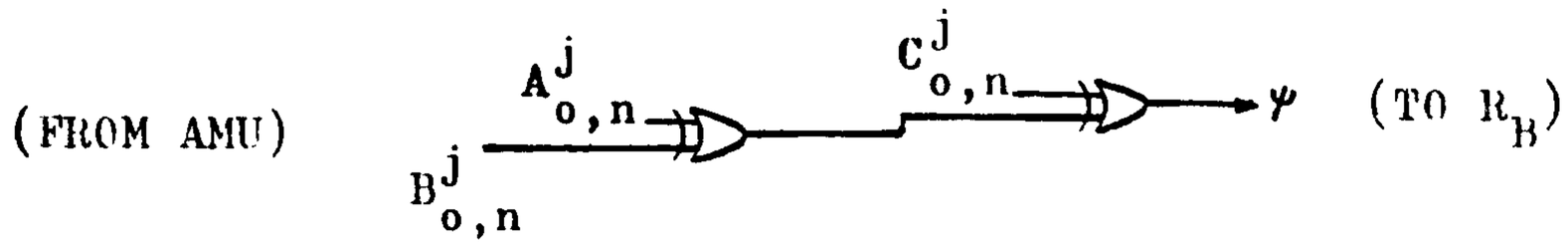


FIG: 7.9.1 THE "END-OVERFLOW CORRECTION UNIT" (EOCU)

As shown in figure 7.7.1, the above End-Overflow Correction function ψ is hard-wired to the D_o input of the 16-bit register R_B. In deriving the above function ψ, it has been implicitly assumed that, at any stage of the "byte-sliced" summation of the compensated partial products of f_n, the input to the register R_B:

$$(\psi, S^j_{o,n}, S^j_{1,n}, \ldots\ldots, S^j_{14,n})$$

should lie within the range of -2 to $(2-2^{-15})$ such that B^{j-1}_n in the next clock cycle would be within the range -1 to $(1-2^{-15})$, that is, in-range. To warrant this, the content of the CBFCS memory should be sufficiently scaled by a factor 2^{-m} as described previously in section 6.7, where m is an integer. The output bits Q_1 and Q_2 of the register R_B are shown in figure 7.7.1 to be wired to the rounding unit RU which will be discussed in the next section.

7.10 THE ROUNDING UNIT (RU):

The rounding unit RU shown in figure 7.1.1 performs the two's complement up-rounding described in section 6.5. Pre-rounding was adopted in the demonstration processor to save an extra clock cycle which would have been required if the necessary roundoff was to be performed at the end of the computation of an output sample y_n. In accomplishing the above pre-rounding, a particular bit of the 16-bit output of the register R_B is set to 1 in the first clock cycle of the 9-clock-cycle operation of the demonstration processor. In the remaining clock cycles, the above mentioned pre-rounding bit will propagate towards the least significant end of the AMU until it finally reaches the position of the most significant bit of the portion of y_n that is to be discarded after rounding.

Table 7.1.1 in section 7.1 showed the above pre-rounding process in the 9-clock-cycle operation of the demonstration processor and figure 7.10.1 shows the circuit configuration of RU.

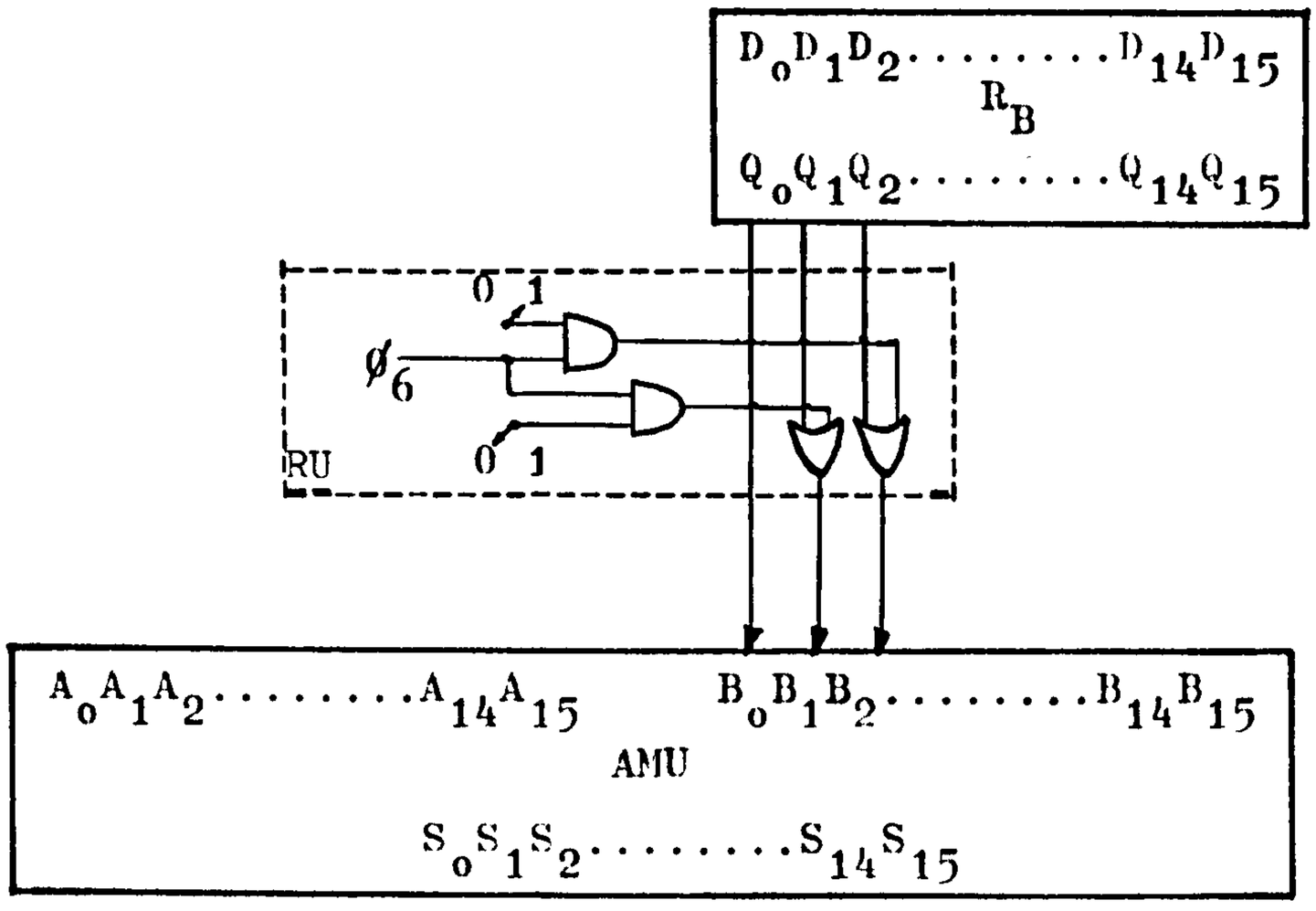


FIG: 7.10.1 LOGIC DIAGRAM AND HARDWARE CONNECTIONS OF THE ROUNDING UNIT RU TO THE AMU AND R_B

In the above hardware implementation of RU, provisions have been made to accommodate the scaling of the content of the CBFCS memory of the demonstration processor by a factor 2^{-m} where $m \in \{1, 2\}$. In the event of scaling the content of the CBFCS memory by 2^{-1}, the Q_1 output bit of R_B will be set to 1 in the first clock cycle via an OR gate controlled by the control signal ϕ_6 shown above. Similarly, ϕ_6 also forces the output bit Q_2 to 1 via an OR gate when the content of the CBFCS memory is to be scaled by 2^{-2}. Either one of the above scaling factor can be manually selected via the two manual switches shown above which are coupled to their respective OR gates via two AND gates, such that the rounding unit RU can be adopted to either the scaling of the content of the CBFCS memory by 2^{-1} or 2^{-2}. (When m is an integer other than 1 or 2, similar arrangements can then be made to modify RU for a general scaling factor 2^{-m}). An alternative and easier implementation of RU is to use, instead of two OR gates, two D-type flip-flops for bits Q_1 and Q_2, such that the output of either of these flip-flops can be set to 1 in the first clock cycle

to perform the required pre-rounding. The choice of the above two OR gates instead of two D-type flip-flops in the demonstration processor was purely based on the reason of less power consumption in the former case. As in the rest of the demonstration processor, Schottky logic gates were used to implement RU.

7.11 THE COMPENSATION UNITS CU' AND CU":

As previously mentioned in section 7.1, after eight clock cycles the value of f'_n of the compensation function f' (equation 5.2.12) is produced and clocked into the compensation unit CU' at the start of the 9th clock cycle as shown below:

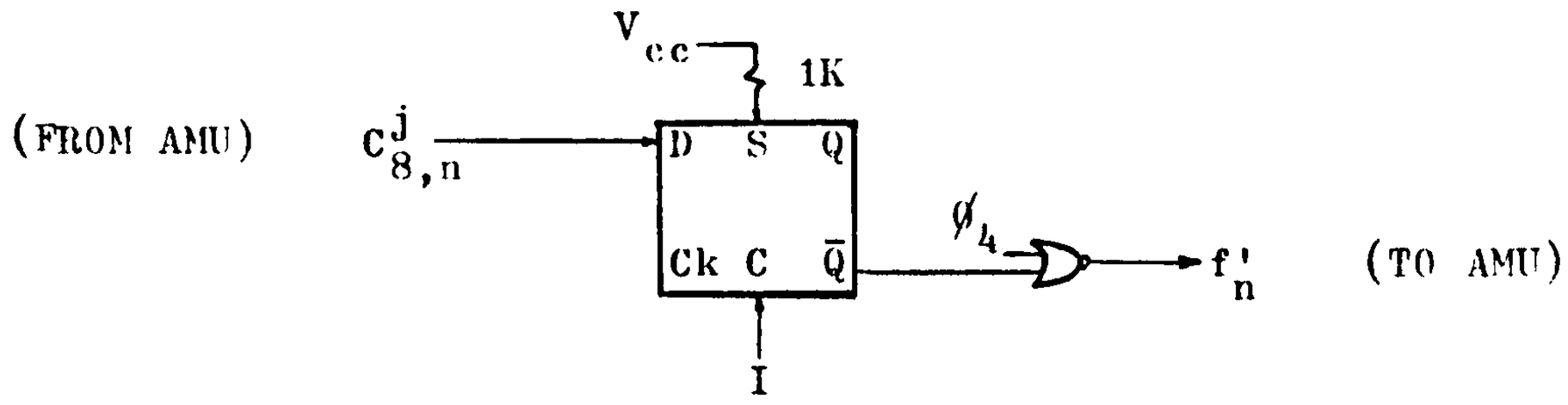


FIG: 7.11.1 COMPENSATION UNIT CU'

The output f'_n of the compensation unit CU' is connected to the AMU as shown in figure 7.7.1. The function f'_n can take on a value of 0 or 2^{-7} depending on the value of $C^1_{8,n}$. The value of f_n is first compensated as shown in table 7.1.1 by the value f'_n in the 9th clock cycle. In the hardware implementation of CU', the right-shift 2^{-7} is hard-wired so that the value of $C^1_{8,n}$ is being weighted and brought forward to be stored in a D-type flip-flop whose output is gated out every 9th clock cycle, through the NOR gate shown above, controlled by the control signal $\emptyset_4$. As shown in figure 7.11.1 above, the input of the compensation unit CU' is hard-wired to the $C^j_{8,n}$ output of the AMU, and since the output of CU' is gated out only in every 9th clock cycle, the input signal to CU' is in effect $C^1_{8,n}$. The D-type flip-flop is initialized by the initialization pulse I when the power is first switched on to ensure correct operations of the compensation unit CU'. The logic function for the output of the compensation unit CU' is $Q.\bar{\emptyset}_4 = \overline{\bar{Q}.\emptyset_4}$.

As given by equation (5.2.14), the output sample y_n of the proposed CBFCS algorithm is

$$y_n = f_n + f'_n + f''_n$$

In the demonstration processor, the value of f_n is first compensated by f'_n, yielding the value $f_n + f'_n$ at the output of the AMU in the 9th clock cycle. At the end of the 9th clock cycle, f''_n is also produced, however, the value of f_n is not compensated by f''_n at this instance. In fact, f_n of the present output sample y_n will be compensated by the corresponding f''_n in the next nine clock cycles when the demonstration processor is computing the next outpnt sample. Therefore, as in the case of CU', the compensation unit CU" has to generate and store the brought forward carry $C^o_{8,n}$ as shown in figure 7.11.2 below:

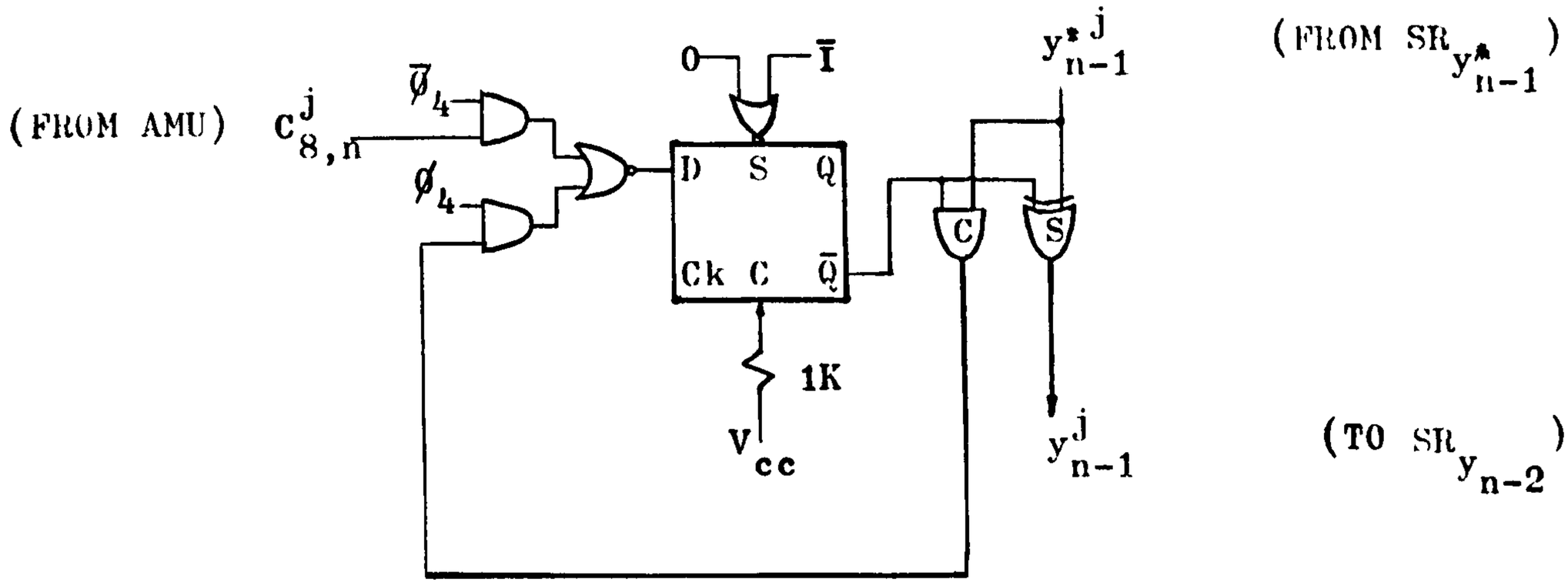


FIG: 7.11.2 COMPENSATION UNIT CU"

The value of f''_n of the compensation function f" (equation 5.2.13 in section 5.2) can take on a value of 0 or 2^{-7} depending on the value of $C^o_{8,n}$. While the value of $f_n + f'_n$ is clocked in parallel into the shift register $SR_{y^*_{n-1}}$ (see a later section of the output port OP), the value of f''_n is also simultaneously clocked into the unit CU" at the start of the next nine clock cycles of the operation of the demonstration processor. In the hardware implementation of CU", the right-shift 2^{-7} is hard-wired and at each data shift in the next nine clock cycles, the content of $SR_{y^*_{n-1}}$ is then shifted serially, with the

least significant bit first, into the compensation unit CU" and bit by bit compensated by the compensation function f" to yield the output

$$y^{j}_{n-1} = (f_{n-1} + f'_{n-1} + f''_{n-1})^{j}$$

which is the serial delayed version of the previous output sample as shown in figure 7.11.2, shifting serially into the shift register $SR_{y_{n-2}}$. One comment is now in order: if the content of the CBFCS memory has been scaled by a factor 2^{-m}, as mentioned before, the value of $f_n+f'_n$ above will be loaded in parallel into the shift register $SR_{y^*_{n-1}}$ with m-bit left-shift hard-wired. Consequently, the content of the shift register $SR_{y^*_{n-1}}$ will now shift serially into the compensation unit CU" starting from the bit, (m+8) places from the most significant end of the shift register $SR_{y^*_{n-1}}$. (See a later section on the output port OP).

In figure 7.11.2, the bit by bit compensation to form y^{j}_{n-1} as described above is performed by a half-adder unit (since f''_{n-1} is a single bit) implemented in hardware in the demonstration processor with a single gate-delay via the AND gate C and the EX-OR gate S, whose outputs are the carry and sum of f''_{n-1} and $f_{n-1}+f'_{n-1}$ respectively as shown. The logic equations and state table of the above half-adder unit are given below:

$\bar{Q}$	y^{*j}_{n-1}	y^{j}_{n-1}	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLE: 7.11.1 STATE TABLE OF THE HALF-ADDER UNIT IN THE COMPENSATION UNIT CU"

LOGIC EQUATIONS:

$$y^{j}_{n-1} = \bar{Q} \oplus y^{*j}_{n-1} \qquad (7.11.1)$$

$$C = \bar{Q} . y^{*j}_{n-1} \qquad (7.11.2)$$

The AND/NOR gate at the input of the D-type flip-flop shown in figure 7.11.2 requires that the $\bar{Q}$ output of the flip-flop to be used in order to form an AND/OR logic function, controlled by the control signals ϕ_4 and $\bar{\phi}_4$, to select the $C^j_{8,n}$ and C inputs via $\bar{\phi}_4$ and ϕ_4 respectively. Furthermore, since the signal $C^j_{8,n}$ from the AMU is selected by $\bar{\phi}_4$ in every 9th clock cycle of the operation of the demonstration processor, the input to the D-type flip-flop in the 9th clock cycle is in effect $C^o_{8,n}$. In the other eight clock cycles, the input C to the D-type flip-flop from the AND gate C, is selected by ϕ_4 to complete the above mentioned bit by bit compensation to form the signal y^j_{n-1}. In the demonstration processor, one package of SN74S74 was used to implement CU' and CU" as shown in figures 7.11.1 and 7.11.2, together with the required Schottky TTL logic gates.

Furthermore, since the compensation of f_n by $f"_n$ requires only one gate delay, it has been performed while the output port (OP) (shown in figure 7.13.1 in section 7.13) is serially shifting out its content. Thus, the design of CU" has made use of some "dead time" of the output port (OP) section of the demonstration processor. (For a definition of the term "dead time", see section 7.13 on the output port (OP) section). This design concept, together with the concept of bringing forward the carry $C^o_{8,n}$ to perform the compensation of f_n by $f"_n$ for the present output sample y_n in the computation time of the next output sample, is in line with the design concept of introducing the register R_A (section 7.5) in the present "psuedo-pipeline" demonstration processor. The above mentioned concepts have enabled the work load of the demonstration processor to be distributed in various sections throughout the entire system in such a way that the processing time of each section has been evened up in order to achieve the shortest possible time required to produce an output sample y_n. Finally, the signal $\overline{0+\bar{I}}$ at the output of the NOR gate connected to the set input S of the D-type flip-flop shown in figure 7.11.2, by-passes the compensation unit CU" by setting the $\bar{Q}$ output of the D-type flip-flop to 0 when saturation arithmetic is initiated in the demonstration processor, and, also initializes the compensation unit CU" when power is first switched on. The signal 0 is generated by the "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) which will be described in the next section.

When the demonstration processor is multiplexed to implement higher order filters, the compensation unit CU" described above has to be multiplexed in a manner as shown in figure 5.4.3 in section 5.4. Figure 7.11.3 shows a multiplexed CU" for an 8th order bandpass filter designed by the design technique described in chapter three. Since four parallel second-order sections (elemental filters) are required to form the above 8th order filter, four D-type flip-flops, viz., D_y, D_w, D_v and D_u, have to be used in the multiplexed CU".

Each D-type flip-flop in turn stores the brought forward carry $C^o_{8,n}$ for the corresponding section-order section currently processed by the now multiplexed demonstration processor. The AND/NOR gate at the input of the 1:4 demultiplexer ($\frac{1}{2}$xSN74S139) shown in figure 7.11.3 requires that the $\bar{Q}$'s (inverted outputs) of the four D-type flip-flops to be used in order to form an AND/OR logic function to select the $C^j_{8,n}$ and C inputs via the control signals $\bar{\emptyset}_4$ and $\emptyset_4$ respectively. Furthermore, since each of the $\bar{Q}$'s of the above D-type flip-flops is processed in turn, a 4:1 multiplexer ($\frac{1}{2}$xSN74S153) is synchronized with the above 1:4 demultiplexer to route the appropriate $\bar{Q}$ to the inputs of the AND gate C and the EX-OR gate S whose other inputs are then connected, via another synchronized 4:1 multiplexer ($\frac{1}{2}$xSN74S153), to the signals

$$(y^{*j}_{n-1},\ w^{*j}_{n-1},\ v^{*j}_{n-1},\ u^{*j}_{n-1})$$

obtained from four structurally identical output ports for the above four second-order sections (as will be discussed in section 7.13 and shown in figure 7.13.2). The output of the above EX-OR is then wired to another synchronized 1:4 demultiplexer ($\frac{1}{2}$xSN74S139) whose output sequentially yields the signals

$$(y^j_{n-1},\ w^j_{n-1},\ v^j_{n-1},\ u^j_{n-1})$$

which are the delayed serial outputs of the above four second-order sections. These delayed outputs are then sent to the appropriate shift registers in the above four output ports as will be shown in figure 7.13.2. The above synchronization of the multiplexers and demultiplexers is achieved by connecting their select inputs to the (Q_1, Q_2) outputs of a divide-by-4 synchronous counter (logic diagram will be shown in figure 7.13.3) such that it sequentially selects the four output ports for the corresponding four D-type flip-flops

in the multiplexed CU". As before, since the signal $C^{j}_{8,n}$ from the AMU is selected by $\bar{\emptyset}_4$ in every 9th clock cycle, the input to each of the above D-type flip-flops during this time is, in effect, its corresponding $C^{o}_{8,n}$. In the other eight cycles, the input C from the AND gate C will be selected instead, by $\emptyset_4$, to complete the bit by bit compensation function of the multiplexed CU" whose operation in conjunction with the output ports for the four parallel second-order sections mentioned previously, will be discussed in section 7.13. Finally, the signals ($\overline{0_y+\bar{I}}$, $\overline{0_w+\bar{I}}$, $\overline{0_v+\bar{I}}$, $\overline{0_u+\bar{I}}$) shown in figure 7.11.3 have the same function as the signal $\overline{0+\bar{I}}$ shown in figure 7.11.2.

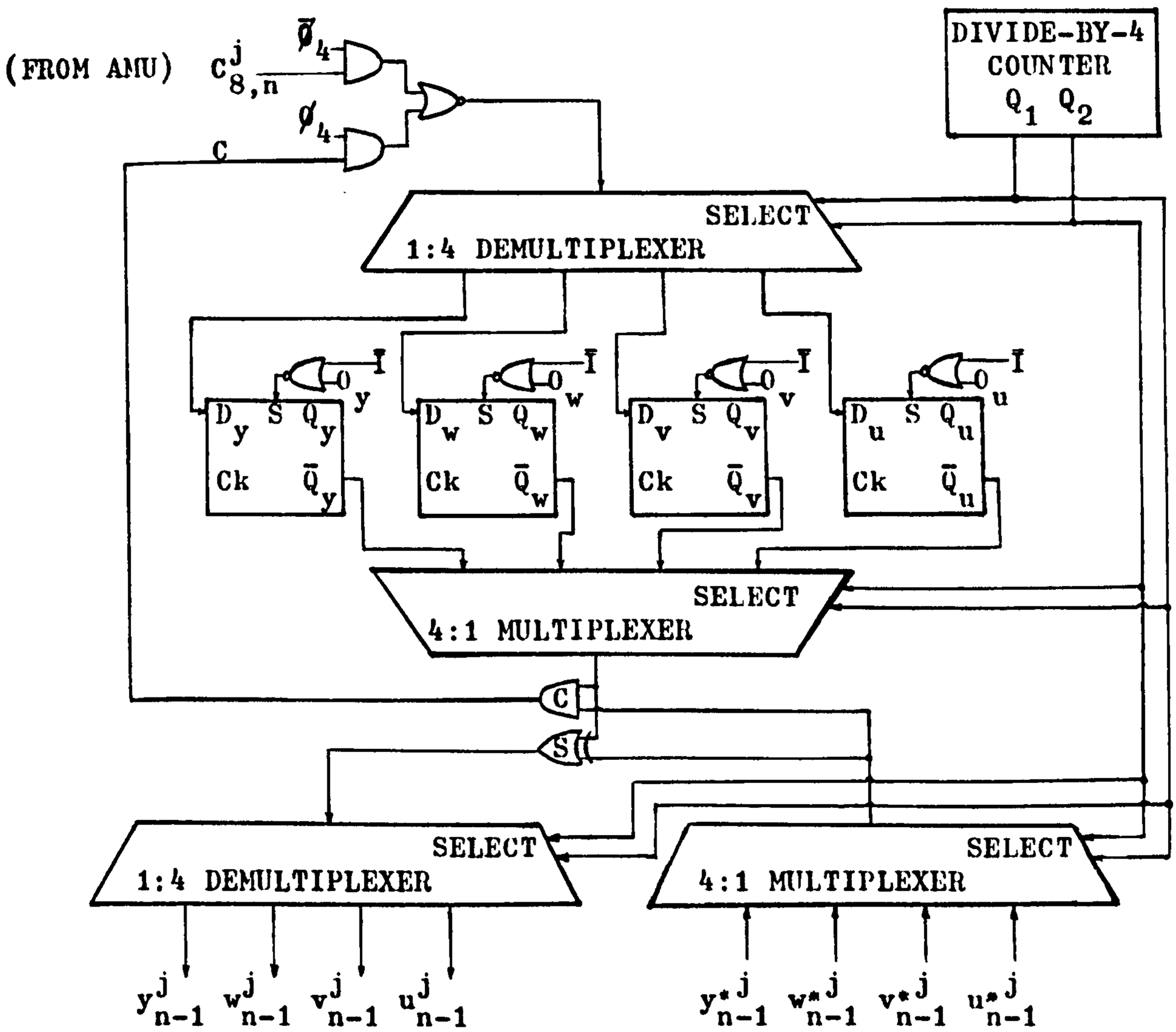


FIG: 7.11.3 LOGIC DIAGRAM SHOWING THE MULTIPLEXED COMPENSATION UNIT CU" FOR A MULTIPLEXED 8TH ORDER FILTER

7.12 THE "OVERFLOW DETECTION AND SATURATION ARITHMETIC UNIT" (ODSAU):

When the value of an output sample y_n has exceeded the dynamic range of the demonstration processor, an out-of-range overflow is said to have occurred. As previously described in section 6.7, such overflows are highly undesirable and result in overflow oscillations discussed in section 6.8. Detection is thus necessary to initiate a type of saturation arithmetic described in section 6.8.1 in order to prevent overflow oscillations. For this purpose, two approaches exist which can be used in the demonstration processor. The first approach entails a conventional technique used in most existing hardware implementations of digital filters. The second approach relies on the same technique used in designing the "End-Overflow Correction Unit" (EOCU) of the demonstration processor. Although the second approach was used in the implementation of the demonstration processor for its simplicity relative to the first approach, both approaches will be discussed and compared below, for the sake of completeness.

In the first approach, an out-of-range overflow is said to have occurred when the carry out of the sign position differs from the carry into the sign position, viz.,

$$\text{OVERFLOW} = C^{o}_{o,n} \oplus C^{o}_{1,n} \qquad (7.12.1)$$

Since in the demonstration processor, the sign bit occupies the most significant bit of one of the four 4-bit standard TTL full adders used to implement the AMU, the carry into this position is not directly available on chip and has to be generated by some external logic gates as follows:

$$C^{o}_{1,n} = A^{o}_{o,n} \oplus B^{o}_{o,n} \oplus S^{o}_{o,n} \qquad (7.12.2)$$

such that,

$$\text{OVERFLOW} = C^{o}_{o,n} \oplus A^{o}_{o,n} \oplus B^{o}_{o,n} \oplus S^{o}_{o,n} \qquad (7.12.3)$$

A positive overflow is further said to have occurred when the sum of two positive inputs to the AMU in the 9th clock cycle is greater than the positive dynamic range of the demonstration processor.

In this case, $C^0_{0,n}=0$ and A^0_n and B^0_n are both positive, while a positive overflow is indicated as follows:

$$\text{POSITIVE OVERFLOW} = \bar{C}^0_{0,n}.C^0_{1,n} \qquad (7.12.4)$$

Similarly, the expression for a negative overflow is:

$$\text{NEGATIVE OVERFLOW} = C^0_{0,n}.\bar{C}^0_{1,n} \qquad (7.12.5)$$

In this case, $C^0_{0,n}=1$ and A^0_n and B^0_n are both negative. Note that,

$$\begin{aligned}\text{OVERFLOW} &= (\text{NEGATIVE OVERFLOW}) + (\text{POSITIVE OVERFLOW})\\ &= C^0_{0,n}.\bar{C}^0_{1,n} + \bar{C}^0_{0,n}.C^0_{1,n}\\ &= C^0_{0,n} \oplus C^0_{1,n}\end{aligned}$$

which is the logic equation for an out-of-range overflow.

The second approach used in the implementation of the demonstration processor follows from a necessary and sufficient condition for an out-of-range overflow, which is the signs of A^0_n and B^0_n must be alike while the sign of S^0_n is different from those of A^0_n and B^0_n, that is,

$$\text{OVERFLOW} = \bar{A}^0_{0,n}.\bar{B}^0_{0,n}.S^0_{0,n} + A^0_{0,n}.B^0_{0,n}.\bar{S}^0_{0,n} \qquad (7.12.6)$$

while the corresponding expression for a positive overflow is:

$$\text{POSITIVE OVERFLOW} = \bar{A}^0_{0,n}.\bar{B}^0_{0,n}.S^0_{0,n} \qquad (7.12.7)$$

and that for a negative is:

$$\text{NEGATIVE OVERFLOW} = A^0_{0,n}.B^0_{0,n}.\bar{S}^0_{0,n} \qquad (7.12.8)$$

Therefore, in comparison, the second approach yields a set of simpler logic equations to indicate an overflow than the first approach. Another reason for not adopting the first approach in the demonstration processor is that the 4-input EX-OR operation in equation (7.12.3), entails a much longer propagation delay compared to that of the 3-input AND operation in equation (7.12.6). This is very important when the demonstration processor is required to operate at a maximum possible speed. Furthermore, the power dissipated by a 4-input EX-OR gate is much higher than that of a 3-input AND gate when Schottky TTL technology has to be used to acquire speed. Having dealt with the choice of the second approach, we shall next consider its modification when the content of the CBFCS memory has been scaled by 2^{-m} where m is an integer.

When the content of the CBFCS memory has been scaled by a factor 2^{-m}, the output of the AMU in the 9th clock cycle (when the depleted sum S_n^o has been produced) has to be re-scaled by a factor 2^m in order to produce the correct output sample y_n. Obviously, this re-scaling of S_n^o by a factor of 2^m has an important bearing on the indication of an overflowed or out-of-range y_n. This is best made clear by way of examples below:

Case 1:

$$\begin{array}{rl} 0.0111000000000000 & \longleftarrow A_n^o \\ +0.0111000000000000 & \longleftarrow B_n^o \\ \hline 0.1110000000000000 & \longleftarrow S_n^o \end{array}$$

In this case, we examine positive overflows with scaling taken into account. Here, A_n^o=0.4375, B_n^o=0.4375 so that S_n^o=0.875 and does not overflow if no re-scaling has to be performed. However, when S_n^o has to be re-scaled by a factor 2^m, say m=1, then the re-scaled S_n^o will become 1.7; that is, a positive overflow will occur.

Case 2:

$$\begin{array}{rl} 1.1001000000000000 & \longleftarrow A_n^o \\ +1.1001000000000000 & \longleftarrow B_n^o \\ \hline 1.0010000000000000 & \longleftarrow S_n^o \end{array}$$

In this case, A_n^o=-0.4375, B_n^o=-0.4375 so that S_n^o=-0.875. As in the last case, S_n^o will overflow when re-scaled by a factor 2^m. When m=1, the re-scaled S_n^o becomes -1.7, that is, a negative overflow occurs.

From the above examples, we notice that when overflows occur due to a re-scaling by 2 (for m=1), the values of the bits $S^o_{o,n}$ and $S^o_{1,n}$ of S^o_n are invariably different. In fact, for positive overflows due to a re-scaling by 2, we have the logic expression:

$$\text{POSITIVE OVERFLOW DUE TO RE-SCALING BY 2} = \bar{S}^o_{o,n}.S^o_{1,n} \qquad (7.12.9)$$

since when $S^o_{1,n}=1$, it represents a magnitude of 0.5.

Similarly, for negative overflows due to a re-scaling by 2, we have the logic expression:

$$\text{NEGATIVE OVERFLOW DUE TO RE-SCALING BY 2} = S^o_{o,n}.\bar{S}^o_{1,n} \qquad (7.12.10)$$

Combining equations (7.12.7) and (7.12.8) with equations (7.12.9) and (7.12.10), we have the overall expressions for positive and negative overflows which take into account a re-scaling by 2 as follows:

$$\text{POSITIVE OVERFLOW } 0^+_1 = \bar{A}^o_{o,n}.\bar{B}^o_{o,n}.S^o_{o,n} + \bar{S}^o_{o,n}.S^o_{1,n} \qquad (7.12.11)$$

$$\text{NEGATIVE OVERFLOW } 0^-_1 = A^o_{o,n}.B^o_{o,n}.\bar{S}^o_{o,n} + S^o_{o,n}.\bar{S}^o_{1,n} \qquad (7.12.12)$$

Similar expressions can also be obtained for a re-scaling by 2^m where $m>1$. For instance, when m=2 we have,

$$\text{POSITIVE OVERFLOW } 0^+_2 = \bar{A}^o_{o,n}.\bar{B}^o_{o,n}.S^o_{o,n} + \bar{S}^o_{o,n}.S^o_{1,n} + \bar{S}^o_{o,n}.S^o_{2,n}$$

$$= \bar{A}^o_{o,n}.\bar{B}^o_{o,n}.S^o_{o,n} + \bar{S}^o_{o,n}(S^o_{1,n} + S^o_{2,n}) \qquad (7.12.13)$$

$$\text{NEGATIVE OVERFLOW } 0^-_2 = A^o_{o,n}.B^o_{o,n}.\bar{S}^o_{o,n} + S^o_{o,n}.\bar{S}^o_{1,n} + S^o_{o,n}.\bar{S}^o_{2,n}$$

$$= A^o_{o,n}.B^o_{o,n}.\bar{S}^o_{o,n} + S^o_{o,n}(\bar{S}^o_{1,n} + \bar{S}^o_{2,n}) \qquad (7.12.14)$$

So far, we have neglected the effect of the signal $C^o_{8,n}$ on the detection of overflows, since the above detection is performed on S^o_n

only. Now S_n^0 at the output of the AMU in the 9th clock cycle is given by:

$$S_n^0 = r.2^{-8} + f_n + f_n' - C_{8,n}^0.2^{-7} \qquad (7.12.15)$$

as shown in table 7.1.1 in section 7.1 where r is a pre-rounding bit set as described in section 7.10. Noting further that S_n^0 has to be compensated by

$$f_n'' = C_{8,n}^0.2^{-7} \qquad (7.12.16)$$

and then properly re-scaled to form y_n when the content of the CBFCS memory in the demonstration processor has been scaled, we now consider the effect of $C_{8,n}^0$=1 on the limiting cases of a re-scaling by 2 of $S_n^0+f_n''$ as follows:

Case 1:

$$\begin{array}{r} 1.0111111000000000 \leftarrow S_n^0 \\ +\quad\quad 1 \quad\quad\quad\quad \leftarrow f_n'' \\ \hline 1.1000000000000000 \end{array}$$

As explained in the previous section, the compensation of S_n^0 by f_n'' to form y_n is performed in the next nine clock cycles while the detection of an overflowed y_n is performed in the present 9th clock cycle. (The required re-scaling of $S_n^0+f_n''$ is hard-wired as will be described in the next section). In the above example, before the compensation of S_n^0 by f_n'' as shown, the re-scaled value of S_n^0 at the output of the AMU in the 9th clock cycle will be detected as a negative overflow by equation (7.12.12) since 1.0111111000000000 is slightly greater than -0.5. However, after compensation, the above S_n^0 will become $S_n^0+f_n''$ which is -0.5, and even re-scaling this value by 2, no overflow will occur. Therefore, when the detection of an overflowed y_n is to be performed before the compensation of S_n^0 by f_n'', the previous results on overflow detection cannot be applied directly when $C_{8,n}^0$=1 and modifications are necessary. Nevertheless, a further consideration will show that $C_{8,n}^0$=1, in fact, does not affect the present limiting case when the application of saturation arithmetic is taken into account. As discussed in section 6.8.1 previously, the use of saturation arithmetic requires a negatively overflowed y_n to be set to -1 while a positively overflowed y_n to be set to $(1-2^{-7})$ in the

demonstration processor. Hence, upon detection of the negatively overflowed S_n^o before compensation, the value of y_n will be set to -1 by the "Overflowed Detection And Saturation Arithmetic Unit" (ODSAU) whose logic diagram will be shown later in figure 7.13.1. Hence, this saturated value of y_n is the same as that obtained by re-scaling S_n^o, compensated by f_n'', by 2, and $C_{8,n}^o=1$ will therefore not affect the detection of negative overflows by equation (7.12.12). However, this will not be the case for positive overflows as explained below.

Case 2:

$$\begin{array}{r} 0.0111111100000000 \longleftarrow S_n^o \\ + \quad\quad 1 \quad\quad\quad\quad \longleftarrow f_n'' \\ \hline 0.1000000000000000 \end{array}$$

In this case, before the compensation of S_n^o by f_n'', the re-scaled value of S_n^o will not be detected by equation (7.12.11) as a positive overflow as 0.0111111100000000 is slightly less than 0.5. However, after compensation, the above S_n^o will become $S_n^o+f_n''$ which is 0.5 and will cause a positive overflow when re-scaled by 2, resulting in a value of -1 instead of the correct value of 1. Hence, we have to modify equation (7.12.11) by detecting the occurence of this limiting case so that overflow detection can be performed before compensating S_n^o by f_n''. The occurence of the above limiting case is indicated by the logic expression:

$$\bar{S}_{0,n}^o.\bar{S}_{1,n}^o.S_{2,n}^o.S_{3,n}^o.S_{4,n}^o.S_{5,n}^o.S_{6,n}^o.S_{7,n}^o.C_{8,n}^o = 1 \qquad (7.12.17)$$

Hence, the overall overflow detection expressions for a re-scaling by 2, that is m=1, become as follows:

$$0_1^+ = \bar{S}_{0,n}^o.\bar{S}_{1,n}^o.S_{2,n}^o.S_{3,n}^o.S_{4,n}^o.S_{5,n}^o.S_{6,n}^o.S_{7,n}^o.C_{8,n}^o +$$

$$\bar{A}_{0,n}^o.\bar{B}_{0,n}^o.S_{0,n}^o + \bar{S}_{0,n}^o.S_{1,n}^o \qquad (7.12.18)$$

and $$0_1^- = A_{0,n}^o.B_{0,n}^o.\bar{S}_{0,n}^o + S_{0,n}^o.\bar{S}_{1,n}^o$$

Furthermore, by replacing each 2-level AND/OR expression by a 2-level NAND/NAND expression (De Morgan's Law), we can then implement equation (7.12.18) using NAND gates only as shown in figure 7.13.1. Similarly,

the overflow detection expressions for a re-scaling by 4, that is m=2, become as follows:

$$0_2^+ = \bar{S}^0_{0,n}.\bar{S}^0_{1,n}.\bar{S}^0_{2,n}.S^0_{3,n}.S^0_{4,n}.S^0_{5,n}S^0_{6,n}.S^0_{7,n}.C^0_{8,n} +$$

$$\bar{A}^0_{0,n}.\bar{B}^0_{0,n}.S^0_{0,n} + \bar{S}^0_{0,n}(S^0_{1,n} + S^0_{2,n}) \qquad (7.12.19)$$

and $$0_2^- = A^0_{0,n}.B^0_{0,n}.\bar{S}^0_{0,n} + S^0_{0,n}(\bar{S}^0_{1,n} + \bar{S}^0_{2,n})$$

As in the case of m=1, by replacing each 2-level AND/OR expression by a 2-level NAND/NAND expression (De Morgan's Law), we can then implement equation (7.12.19) using NAND gates only as shown in figure 7.13.1 in the next section.

Upon detection of an overflow, the "Overflow Detection And Saturation Arithmetic Unit" shown in figure 7.13.1, will initiate a type of saturation arithmetic as explained in section 6.8.1. When implemented in the demonstration processor, this type of saturation arithmetic requires that when an output sample y_n has overflowed, it is then saturated to its maximum dynamic range value with respect to the sign of the overflow. Consequently, the signal

$$0^+ \epsilon \{0_1^+,\ 0_2^+\} \qquad (7.12.19)$$

which indicates positive overflows for m=1 or m=2, sets $SR_{y^*_{n-1}}$ to the value of $(1-2^{-7})$ as will be explained in the next section. Similarly, the signal

$$0^- \epsilon \{0_1^-,\ 0_2^-\} \qquad (7.12.20)$$

which indicates negative overflows for m=1 or m=2, sets $SR_{y^*_{n-1}}$ to the value of -1 as shown in figure 7.13.1. The signal

$$0 = 0^+ + 0^- \qquad (7.12.21)$$

clears the compensation unit CU" in the case of an overflow as shown in figures 7.11.2 and 7.13.1 and described in section 7.11, for the correct operation of the above mentioned saturation arithmetic.

7.13 THE OUTPUT PORT (OP):

The output port (OP) of the demonstration processor consists of the "Overflow Detection And Saturation Arithmetic Unit" (ODSAU), the "Scaling Correction Unit" (SCU) and the shift registers $SR_{y^*_{n-1}}$ and $SR_{y_{n-2}}$ shown as integral parts in figure 7.1.1 in section 7.1.

Figure 7.13.1, however, depicts the actual logic design of the various units above, together with the compensation unit CU" described in section 7.11. The outputs y^j_{n-1} and y^j_{n-2} shown in figure 7.13.1 are used to address the CBFCS memory as shown in figures 7.4.1 and 7.4.2 previously. As depicted in figure 7.13.1, the shift register $SR_{y^*_{n-1}}$ is made up of ten D-type flip-flops whose inputs are wired to the outputs of ten AND/NOR gates controlled by the control signals ϕ_4 and $\overline{\phi}_4$ such that the following ten bits:

$$(S^0_{0,n},\ S^0_{1,n},\ S^0_{2,n},\ S^0_{3,n},\ \ldots.,\ S^0_{6,n},\ S^0_{7,n},\ S^0_{8,n},\ S^0_{9,n})$$

from the most significant end of the 16-bit output of the AMU in every 9th clock cycle are selected by $\overline{\phi}_4$ and loaded in parallel into $SR_{y^*_{n-1}}$ at the start of the first of the next nine clock cycles, when the delayed version y_{n-1} of the previous output sample is serially shifted (via the shifting control signal ϕ_4) into the shift register $SR_{y_{n-2}}$ (9-bit long) whose content is also serially shifted out every nine clock cycles in the operation of the demonstration processor.

The operation of the output port (OP) in conjunction with the compensation unit CU", however, was already described in section 7.11 and, consequently, will not be repeated here. The re-scaling of the output of the demonstration processor, due to a scaling of the content of the CBFCS memory as discussed previously, is performed by the "Scaling Correction Unit" (SCU) which is made up of the three AND/NOR logic gates (shaded) and shown together as an integral part of the output port (OP) in figure 7.13.1. The "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) consists of the two D-type flip-flops, viz., D_A and D_B and the combinatorial logic circuit that generates the signals 0, 0^+ and 0^- of equations (7.12.19)-(7.12.21) in the last section. The above combinatorial logic circuit is implemented in NAND gates as shown in figure 7.13.1. The two D-type

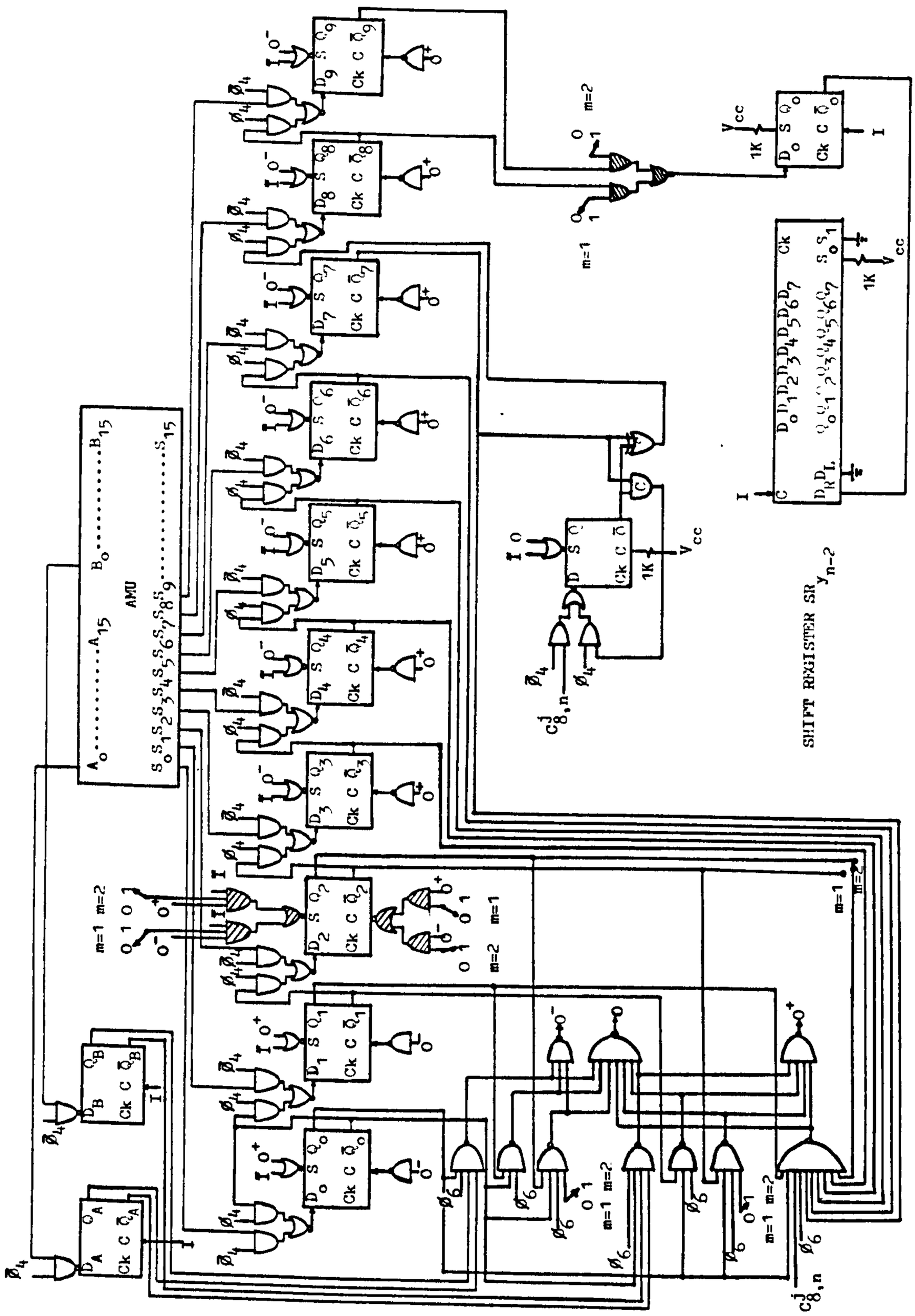


FIG: 7.13.1 LOGIC DIAGRAM SHOWING THE STRUCTURE OF THE OUTPUT PORT (OP).

flip-flops, D_A and D_B, are for storing the most significant bits, that is the sign bits $A^o_{o,n}$ and $B^o_{o,n}$, of the inputs to the AMU in every 9th clock cycle of the operation of the demonstration processor, for the purpose of generating the above mentioned signals 0, 0^+ and 0^-.

In the hardware implementation of the output port (OP) shown in figure 7.13.1, the output y^j_{n-1} is either gated serially out of the least or the second least significant D-type flip-flop of the shift register $SR_{y^*_{n-1}}$ into the shift register $SR_{y_{n-2}}$, by one of the AND/NOR gates of the "Scaling Correction Unit" (SCU). The re-scaling of the output of the demonstration processor by 4 (i.e. m=2) or by 2 (i.e. m=1), corresponding to gating the output of the least or the second least significant D-type flip-flop of the shift register $SR_{y^*_{n-1}}$, is selected by setting the appropriate manual switches of the "Scaling Correction Unit" (SCU) at the start of the operation of the demonstration processor. The set input S and the clear input C of the third most significant D-type flip-flop of the shift register $SR_{y^*_{n-1}}$ are connected to the other two AND/NOR gates of the "Scaling Correction Unit" (SCU), so that the signals 0^+ or 0^- from the "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) can be pre-selected, via the manual switches shown corresponding to a re-scaling by 2 or by 4, to clear or set the D-type flip-flop when saturation arithmetic is initiated. Hence upon detection of overflow of an output sample y_n, saturation arithmetic is initiated, and for a re-scaling by 4 (i.e. m=2), the $\bar{Q}$'s of the ten D-type flip-flops of the shift register $SR_{y_{n-1}}$ are appropriately set or cleared by the signal 0^- or 0^+ into the state (1100000000) or (0011111111) for negative or positive overflows respectively. For a re-scaling by 2 (i.e. m=1), the above $\bar{Q}$'s are then set or cleared appropriately by the signal 0^- or 0^+ into the state (1110000000) or (0001111111) for negative or positive overflows respectively. In the hardware implementation of the "Overflow Detection And Saturation Arithmetic Unit" (ODSAU), the selection of the signals 0^+_1 and 0^-_1, or 0^+_2 and 0^-_2 in equations (7.12.19) and (7.12.20), corresponding to the scaling of the content of the CBFCS memory by 2 (i.e. m=1) or by 4 (i.e.m=2), is done via the manual switches shown in figure 7.13.1 at the start of the operation of the demonstration processor. The initialization pulse I has been used as shown in figure 7.13.1 to initialize the

output port (OP) when power is first switched on.

At this juncture, some comments are now in order. Since the above overflow detection and initialization of saturation arithmetic are performed in every first clock cycle of the 9-clock-cycle operation of the demonstration processor, the control signal ϕ_6 has been included in the hardware implementation of equations (7.12.19)-(7.12.21). That this has been feasible is due to the fact that there is a certain amount of "dead time" available in the output port (OP) which allows the execution of the above processes. (A certain amount of "dead time" is said to be available in the output port (OP) of the demonstration processor if the propagation delay of the shift register $SR_{y^*_{n-1}}$ is shorter than the clock period).

As in the case of input data, the serial outputs y^j_{n-1} and y^j_{n-2} of the output port (OP) are such that their most significant ends are each padded with an extra bit. This extra padding bit of each of the above signals, has the same logical value as that of the sign bit of the signal it associates with. This is a direct consequence of the "psuedo-pipeline" structure of the demonstration processor.

In the demonstration processor, the dynamic range has been limited to eight-bit accuracy. Consequently, only eight of the ten bits of the 16-bit output of the AMU which are loaded in parallel into the output port (OP), have been processed so far to yield an eight-bit output sample of the demonstration processor. However, if the full sixteen bits of the output of the AMU are processed in the same way as the above ten bits, a "~~psuedo~~ pseudo-sixteen-bit" output sample of the demonstration processor can then be achieved. This then represents an improvement in the accuracy of the output sample. The AMU can only provide the above "~~psuedo~~ pseudo-sixteen-bit" accuracy since, in implementing recursive difference equations, the dynamic range of the signals used to address the CBFCS memory has only eight-bit accuracy. On the other hand, when the demonstration processor is used to implement non-recursive difference equations, then full 16-bit accuracy can be obtained from the output of the 16-bit AMU.

When the demonstration processor is multiplexed to implement higher order filters designed by the proposed design technique discussed in chapter three, more than one output port (OP) have to be used. For instance, if an 8th order bandpass filter is to be

implemented by multiplexing the AMU of the demonstration processor, four output ports have to be used which are sequentially selected via a 1:4 demultiplexer (SN74S139's) as shown in figure 7.13.2. The 1:4 demultiplexer routes the previously described ten bits of the 16-bit output of the now multiplexed AMU to the individual output ports, by addressing its select inputs with the outputs (Q_1, Q_2) of a synchronous divide-by-4 counter shown in figure 7.13.3. As already mentioned in sections 7.4 and 7.11, this 1:4 demultiplexer at the output of the now multiplexed AMU is also synchronized with the rest of the 4:1 multiplexers and 1:4 demultiplexer described in the above sections, via the same divide-by-4 synchronous counter whose clock input is connected to the control signal $\emptyset_5$ as shown in figure 7.13.3.

The above four output ports are identical in their logic structures and contain the shift registers:

$$(SR_{y^*_{n-1}}, SR_{w^*_{n-1}}, SR_{v^*_{n-1}}, SR_{u^*_{n-1}})$$

which are connected to the multiplexed AMU via the above mentioned 1:4 demultiplexer, and to the multiplexed CU" via a 4:1 multiplexer as described in section 7.11 previously. Corresponding to the above shift registers, there are also contained in the respective output ports, the shift registers:

$$(SR_{y_{n-2}}, SR_{w_{n-2}}, SR_{v_{n-2}}, SR_{u_{n-2}})$$

which are connected to the output of the multiplexed CU" via a 1:4 demultiplexer as described in section 7.11. In addition, each output port has its own "Scaling Correction Unit" (SCU) and "Overflow Detection And Saturation Arithmetic Unit" (ODSAU) whose operations were discussed in the early part of this section, and are, consequently, not included in figure 7.13.2. The operations of the above four output ports in conjunction with the multiplexed CU" and the input port are summarized below for the correct operation of the multiplexed AMU:

FIRST 9 CLOCK CYCLES:

(a) Parallelly load the input sample x_n into the recirculating shift register SR_{x_n} in the first clock cycle. Its content is then right-shifted serially, with the least significant bit leading, in the remaining eight clock cycles. (see section 7.3).

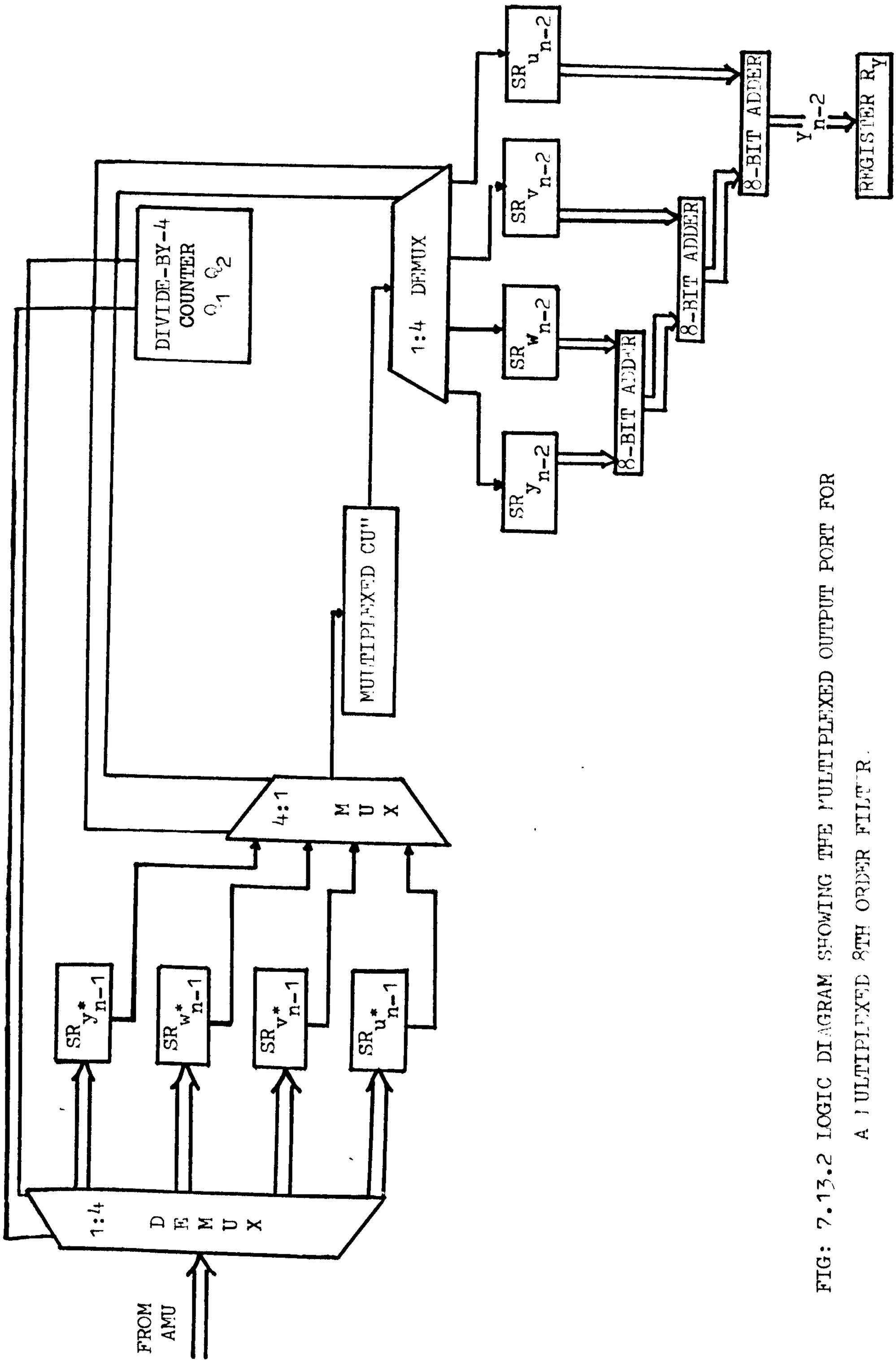


FIG: 7.13.2 LOGIC DIAGRAM SHOWING THE MULTIPLEXED OUTPUT PORT FOR A MULTIPLEXED 8TH ORDER FILTER.

(b) Initialize the recirculating shift register $SR_{x_{n-1}}$ and inhibit its data input by taking its control $\emptyset$ high in the given nine clock cycles. During these nine clock cycles, its content is right-shifted out serially, with the least significant bit leading. (see section 7.3 and figure 7.3.3).

(c) Initialize the shift register $SR_{u^*_{n-1}}$ and $SR_{u_{n-2}}$ in the first clock cycle, and inhibit their clocking in the remaining eight clock cycles.

(d) Initialize and inhibit clocking of the shift registers $SR_{y^*_{n-1}}$ and $SR_{y_{n-2}}$ in the first clock cycle. Clocking of these two shift registers is resumed in the remaining eight clock cycles.

(e) Clocking of the shift registers $SR_{w^*_{n-1}}$, $SR_{w_{n-2}}$, $SR_{v^*_{n-1}}$ and $SR_{v_{n-2}}$ of the other two output ports is inhibited during the given nine clock cycles.

SECOND 9 CLOCK CYCLES:

(a) Parallelly load the content of the recirculating shift register SR_{x_n} in the 9th clock cycle of the previous nine clock cycles (which is still x_n) into itself again in the first clock cycle, and recirculate its present content in the remaining eight clock cycles.

(b) Recirculate the content of the recirculating shift register $SR_{x_{n-1}}$ and inhibit its data input by taking its control $\emptyset$ high in the given nine clock cycles.

(c) Parallelly load the output of the AMU and the signal $C^j_{8,n}$ in the 9th clock cycle of the previous nine clock cycles, into the shift register $SR_{y^*_{n-1}}$ and the multiplexed CU" (see figure 7.11.3 and 7.13.2) in the first clock cycle, and inhibit clocking of the shift registers $SR_{y^*_{n-1}}$ and $SR_{y_{n-2}}$ in the remaining eight clock cycles.

(d) Inhibit clocking of the shift registers $SR_{w^*_{n-1}}$ and $SR_{w_{n-2}}$ in the first clock cycle. Clocking of these two shift registers is resumed in the remaining eight clock cycles.

(e) Clocking of the shift registers $SR_{v^*_{n-1}}$, $SR_{v_{n-2}}$, $SR_{u^*_{n-1}}$ and $SR_{u_{n-2}}$ of the other two output ports is inhibited during the given nine clock cycles.

THIRD 9 CLOCK CYCLES:

(a) Parallelly load the content of the recirculating shift register SR_{x_n} in the 9th clock cycle of the previous nine clock cycles (which is still x_n) into itself again in the first clock cycle, and recirculate its present content in the remaining eight clock cycles.

(b) Recirculate the content of the recirculating shift register $SR_{x_{n-1}}$ and inhibit its data input by taking its control ϕ high in the given nine clock cycles.

(c) Parallelly load the output of the AMU and the signal $C^j_{8,n}$ in the 9th clock cycle of the previous nine clock cycles into the shift register $SR_{w^*_{n-1}}$ and the multiplexed CU" in the first clock cycle, and inhibit clocking of the shift registers $SR_{w^*_{n-1}}$ and $SR_{w_{n-2}}$ in the remaining eight clock cycles.

(d) Inhibit clocking of the shift registers $SR_{v^*_{n-1}}$ and $SR_{v_{n-2}}$ in the first clock cycle. Clocking of these two shift registers is resumed in the remaining eight clock cycles.

(e) Clocking of the shift registers $SR_{u^*_{n-1}}$, $SR_{u_{n-2}}$ $SR_{y^*_{n-1}}$ and $SR_{y_{n-2}}$ of the other two output ports is inhibited during the given nine clock cycles.

FOURTH 9 CLOCK CYCLES:

(a) Parallelly load the content of the recirculating shift register SR_{x_n} in the 9th clock cycle of the previous nine clock cycles (which is still x_n) into itself again in the first clock cycle, and recirculate its present content in the remaining eight clock cycles.

(b) Right-shift out the content of the recirculating shift register $SR_{x_{n-1}}$ whose data input has been activated by its control signal ϕ which is high in the given nine clock cycles. Hence, the content of the recirculating shift register (which is still x_n) is then serially right-shifted into the recirculating shift register $SR_{x_{n-1}}$ in the given nine clock cycles. (Note that the sample x_n will become x_{n-1} in the next nine clock cycles).

(c) Parallelly load the output of the AMU and the signal $C^j_{8,n}$ in the 9th clock cycle of the previous nine clock cycles into the shift register $SR_{v^*_{n-1}}$ and the multiplexed CU" in the first clock cycle, and inhibit clocking of the shift registers $SR_{v^*_{n-1}}$ and $SR_{v_{n-2}}$ in the remaining eight clock cycles.

(d) Inhibit clocking of the shift registers $SR_{u^*_{n-1}}$ and $SR_{u_{n-2}}$ in the first clock cycle. Clocking of these two shift registers is resumed in the remaining eight clock cycles.

(e) Clocking of the shift registers $SR_{y^*_{n-1}}$, $SR_{y_{n-2}}$, $SR_{w^*_{n-1}}$ and $SR_{w_{n-2}}$ of the other two output ports is inhibited during the given nine clock cycles.

FIFTH 9 CLOCK CYCLES:

(a) Parallelly load a new input sample x_n into the recirculating shift register SR_{x_n} in the first clock cycle. Its content is then right-shifted serially, with the least significant bit leading, in the remaining eight clock cycles. This step is the same as step (a) of the first nine clock cycles, hemce, the operation of the multiplexed demonstration processor repeats every 36 clock cycles for an 8th order bandpass filter.

(b) Recirculate the content of the recirculating shift register $SR_{x_{n-1}}$ and inhibit its data input by taking its control $\emptyset$ high in the given nine clock cycles.

(c) Parallelly load the output of the AMU and the signal $C^j_{8,n}$ in the 9th clock cycle of the previous nine clock cycles, into the shift register $SR_{u^*_{n-1}}$ and the multiplexed CU" in the first clock cycle, and inhibit clocking of the shift registers $SR_{u^*_{n-1}}$ and $SR_{u_{n-2}}$ in the remaining eight clock cycles.

(d) Inhibit clocking of the shift registers $SR_{y^*_{n-1}}$ and $SR_{y_{n-2}}$ in the first clock cycle. Clocking of these two shift registers is resumed in the remaining eight clock cycles.

(e) Clocking of the shift registers $SR_{w^*_{n-1}}$, $SR_{w_{n-2}}$, $SR_{v^*_{n-1}}$ and $SR_{v_{n-2}}$ of the other two output ports is inhibited during the given nine clock cycles.

Hence, as described above, the operation of the multiplexed demonstration processor repeats every 36 clock cycles while the required clock inputs to all shift registers in the input and output ports are tabulated in table 7.13.1. The delayed version of the output of the multiplexed 8th order bandpass filter is shown in figure 7.13.2 as Y_{n-2} and given by

$$Y_{n-2} = y_{n-2} + w_{n-2} + v_{n-2} + u_{n-2} \qquad (7.13.1)$$

Equation (7.13.1) is computed with the three conventional 8-bit TTL full adders as shown in figure 7.13.2. The summation at the last adder stage is performed in the first clock cycle of the fifth nine clock cycles, and the sum Y_{n-2} is then clocked into the register R_Y in parallel at the start of the next clock cycle.

OUTPUT OF SYNCHRONOUS DIVIDE-BY-4 COUNTER : Q_2, Q_1. (FIG: 7.13.3)	CLOCK INPUTS TO SHIFT REGISTERS IN INPUT AND OUTPUT PORTS	SHIFT REGISTERS IN INPUT AND OUTPUT PORTS
0 0	$\bar{Q}_1.\bar{Q}_2.Ck = Ck_1$	$SR_{y^*_{n-1}}$, $SR_{y_{n-2}}$
0 1	$Q_1.\bar{Q}_2.Ck = Ck_2$	$SR_{w^*_{n-1}}$, $SR_{w_{n-2}}$
1 1	$Q_1.Q_2.Ck = Ck_3$	$SR_{v^*_{n-1}}$, $SR_{v_{n-2}}$
1 0	$\bar{Q}_1.Q_2.Ck = Ck_4$	$SR_{u^*_{n-1}}$, $SR_{u_{n-2}}$
X X	Ck	SR_{x_n} , $SR_{x_{n-1}}$

TABLE: 7.13.1 CLOCK INPUT SIGNALS TO ALL SHIFT REGISTERS IN INPUT & OUTPUT PORTS

Figure 7.13.3, shows the clock signals to the four output ports.

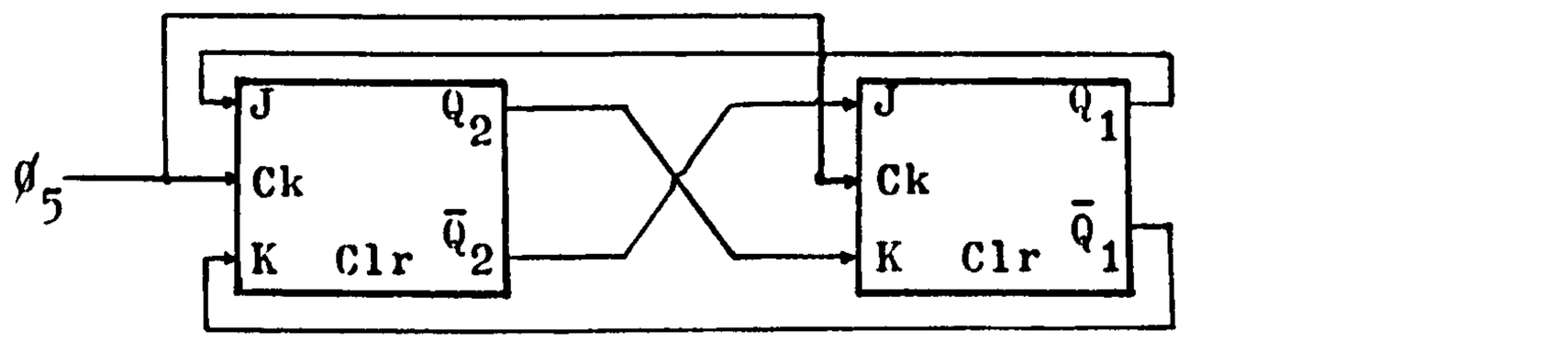


FIG: 7.13.3 GENERATION OF THE CLOCK SIGNALS Ck_1, Ck_2, Ck_3 & Ck_4.

7.14 THE SELECTIVE ROUNDING UNIT (SRU):

In section 6.8 in chapter six, we mentioned the use of a "selective rounding scheme" to eliminate limit cycle behaviours. In this scheme, in addition to injecting random rounding noise, certain output states of the demonstration processor are selectively forbidden. In this section, we shall describe its hardware implementation by modifying the compensation unit CU" and the shift register $SR_{y^*_{n-1}}$ described in sections 7.11 and 7.13 respectively.

Figure 7.14.1 below shows the necessary modifications of the compensation unit CU" while figure 7.14.2 shows the modifications of the shift register $SR_{y^*_{n-1}}$.

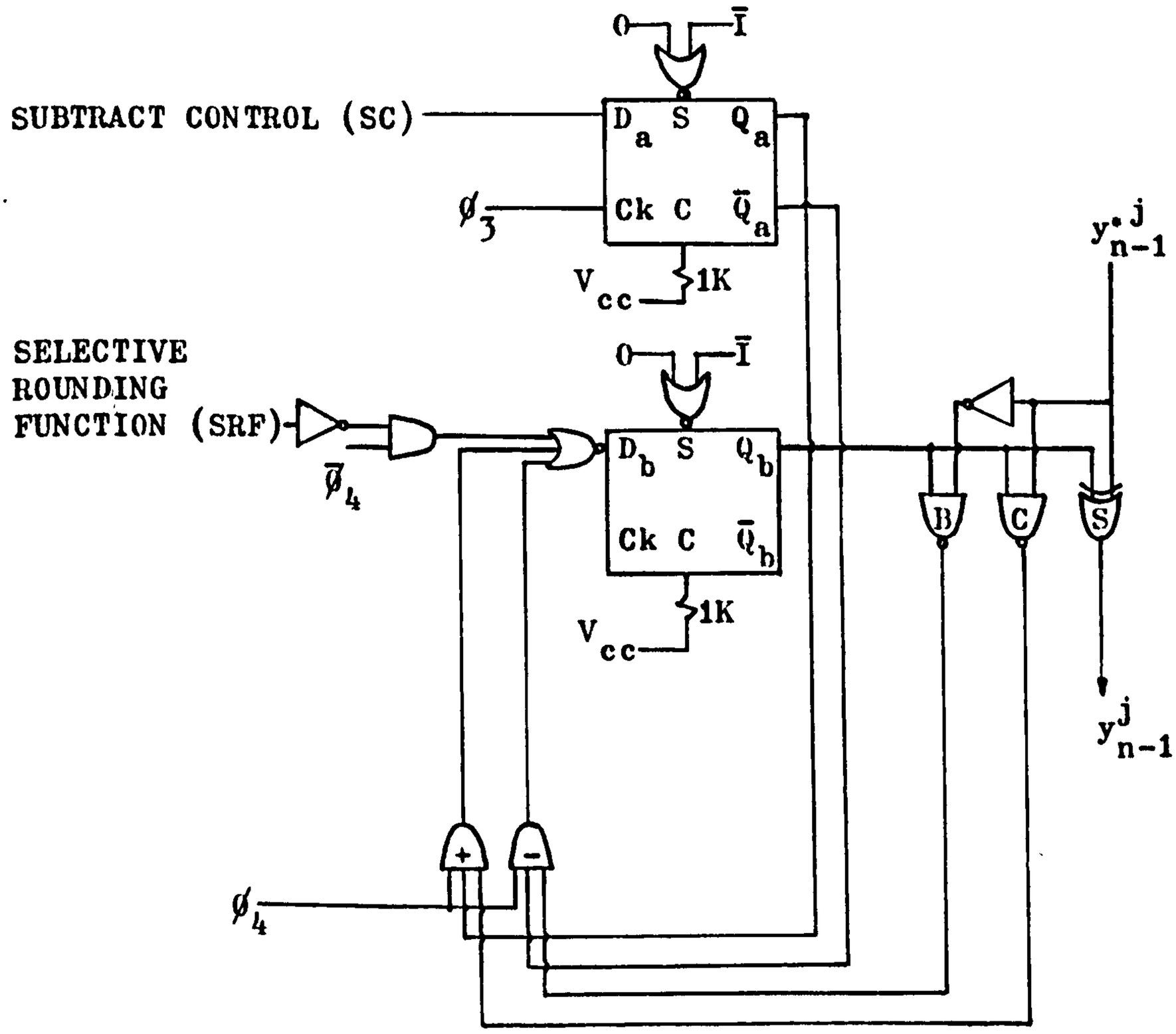


FIG: 7.14.1 LOGIC DIAGRAM SHOWING THE MODIFICATIONS OF THE COMPENSATION UNIT CU" IN THE IMPLEMENTATION OF THE SELECTIVE ROUNDING UNIT (SRU)

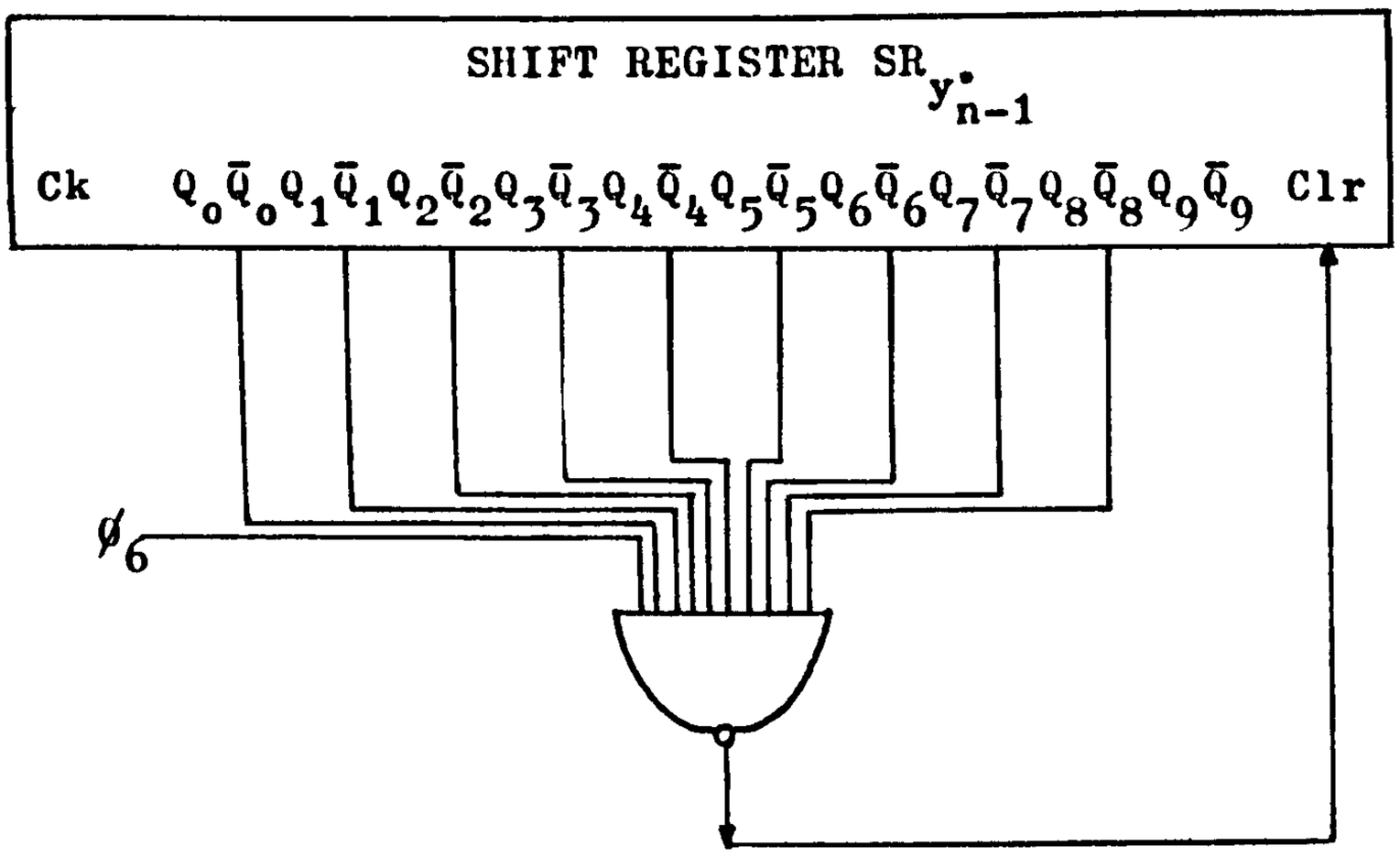


FIG: 7.14.2 LOGIC DIAGRAM SHOWING THE MODIFICATIONS OF THE SHIFT REGISTER SR y^{*}_{n-1} IN THE IMPLEMENTATION OF THE SELECTIVE ROUNDING UNIT (SRU)

Figure 7.14.1 shows the modified compensation unit CU" which is now made up of a half-adder and a half-subtractor units whose logic state tables and diagrams are as follows.

Q	y^{j}_{n-1}	y^{j}_{n-1}	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

HALF-ADDER

Q	y^{j}_{n-1}	y^{j}_{n-1}	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

HALF-SUBTRACTOR

TABLES: 7.14.1a & b. STATE TABLES OF THE HALF-ADDER AND SUBTRACTOR OF THE SELECTIVE ROUNDING UNIT (SRU)

The D-type flip-flop D_b and the EX-OR gate S multiplex as shown in

figure 7.14.1 to form the above half-adder and half-subtractor together with the NAND gates C and B respectively. The relevant logic equations are:

HALF-ADDER: $$y_{n-1}^{j} = Q \oplus y_{n-1}^{j} \qquad (7.14.1)$$

$$C = Q.y_{n-1}^{j} \qquad (7.14.2)$$

HALF-SUBTRACTOR: $$y_{n-1}^{j} = Q \oplus y_{n-1}^{j} \qquad (7.14.3)$$

$$B = Q.\bar{y}_{n-1}^{j} \qquad (7.14.4)$$

The D-type flip-flop D_a provides the necessary subtract and add signals to the D-type flip-flop D_b throughout the nine clock cycles of the operation of the demonstration processor. When the subtract control (SC) at the input of the D-type flip-flop is activated in the 9th clock cycle, it is then clocked into the flip-flop by the control signal ϕ_3 and output of the flip-flop then provides the above subtract signal to the "-" AND gate for the next nine clock cycles. When the subtract control (SC) is not activated, then an add signal is issued instead. In this way, the selective rounding unit can therefore add or subtract random noise, according to the selective rounding function (SRF), to the operations of the demonstration processor. Furthermore, this random noise is superimposed on the roundoff error produced by the two's complement up-rounding unit described in section 7.10, so that the resultant roundoff error is also random. (The rounding unit (RU) described in section 7.10 therefore works in conjunction with the selective rounding unit (SRU) under the selective rounding scheme described in this section). An experimentally verified selective rounding function (SRF) used under the selective rounding scheme is given by the logic equation:

$$SRF = \overline{C_{8,n}^{j} \oplus \overline{S_{o,n}^{j}.S_{8,n}^{j}}} \qquad (7.14.5)$$

Since this selective rounding function is controlled by the control function $\bar{\phi}_4$, its effective value is therefore:

$$SRF.\bar{\emptyset}_4 = \overline{C^o_{8,n} \oplus \overline{S^o_{o,n}.S^o_{8,n}}} \qquad (7.14.6)$$

and its Karnaugh map and the functional table of the modified compensation unit CU" are as follows:

VARIABLES $S^o_{o,n}$ $S^o_{8,n}$ $C^o_{8,n}$	FUNCTION: ADD SRF.$\bar{\emptyset}_4$
0 0 0	1
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

TABLE: 7.14.2 FUNCTIONAL TABLE OF THE MODIFIED COMPENSATION UNIT CU" IN THE SELECTIVE ROUNDING SCHEME

$C^o_{8,n}$ \ $S^o_{o,n}.S^o_{8,n}$	00	01	11	10
0	1	1	0	1
1	0	0	1	0

TABLE: 7.14.3 KARNAUGH MAP OF SRF.$\bar{\emptyset}_4$

To complete the selective rounding scheme, the output state (1.1111111) of the demonstration processor was forbidden in the experimental verification of it for a second-order resonator which had a limit cycle behaviour under zero-input condition when ordinary rounding by the rounding unit (RU) described in section 7.10 was used alone. Figure 7.14.2 shows a 10-input NAND gate used to detect the above forbidden state and cleared the shift register $SR_{y^*_{n-1}}$ such that (1.1111111) would not be included in the output of the processor.

Furthermore, figure 7.14.2 only shows the detection scheme of the above forbidden state for a scaling of the content of the CBFCS memory by a factor of 2. Based on the same idea, a slight modification of the circuit shown in figure 7.14.2 would then allow for the scaling of the content of the CBFCS memory by a factor other than 2. In addition, when the content of the CBFCS memory has been scaled by a factor 2^{-m}, the content of the shift register $SR_{y^*_{n-1}}$ will then shift serially into the modified compensation unit CU" in the selective rounding unit, starting from the bit (m+8) places from the most significant end of the shift register $SR_{y^*_{n-1}}$ as previously described in sections 7.11 and 7.13.

For the selective rounding function SRF described by equation (7.14.6) previously, a random error sequence whose elements are members of the error set:

$$\{2^{-M}+2^{-(M-1)},\ -2^{-M},\ 2^{-M}\}$$

results. This is then the random roundoff error set of the selective rounding scheme. This, on the other hand, can also be viewed as a random noise of one bit, whose magnitude is $2^{-(M-1)}$, superimposed on the roundoff error produced by the roundoff unit (RU) described in section 7.10. This one-bit random noise is added onto the roundoff error of the above rounding unit (RU) via the modified compensation unit CU" shown in figure 7.14.1 by keeping the subtract control (SC) to the D-type flip-flop D_a low throughout the operation of the selective rounding unit (SRU). The experimental result and discussion of applying the above selective rounding function and imposing the forbidden state (1.1111111) in the selective rounding scheme, will be presented in the next chapter for a second-order resonator which, as mentioned previously, had a limit cycle behaviour.

Having discussed and described in detail the logic designs of various functional units of the demonstration processor and its multiplexing to implement higher-order filters, we shall discuss in the next section, the interfacing of the demonstration processor to an analogue-to-digital converter (A/D) and a digital-to-analogue converter (D/A) specially designed for the demonstration processor to process analogue signals.

7.15 INTERFACING THE DEMONSTRATION PROCESSOR:

7.15.1 INTRODUCTION:

Figure 7.15.1 below shows the arrangement used for processing analogue signals with the demonstration processor.

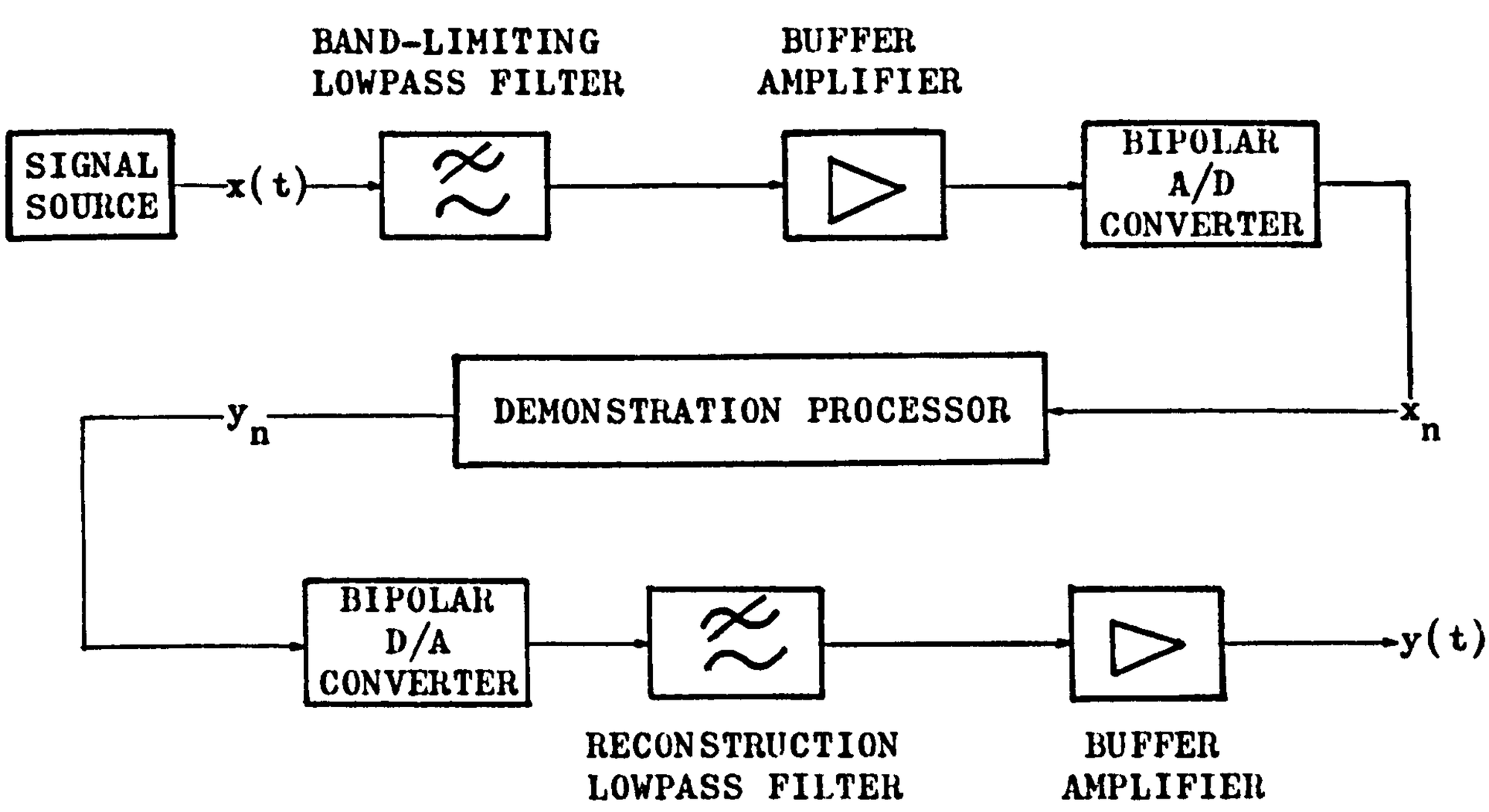


FIG: 7.15.1 ARRANGEMENT USED FOR PROCESSING ANALOGUE SIGNALS WITH THE DEMONSTRATION PROCESSOR

7.15.2 THE BAND-LIMITING LOWPASS FILTER AND BUFFER AMPLIFIER:

As a pre-requisite due to the sampling theorem, the analogue input signal x(t) is first band-limited before sampling so as to avoid or minimize frequency aliasing effects. The required band-limiting of x(t) is performed by the band-limiting active lowpass filter shown in figure 7.15.2, with a cutoff frequency $\frac{1}{2}f_s$ where f_s is the sampling frequency. In making frequency spot checks, the signal x(t) was then generated by a sinusoidal signal source, so that the first-order active lowpass filter used was found to be adequate for the band-limiting purpose. The first 741 operational amplifier shown in figure 7.15.2 is wired as a unity gain follower which is then cascaded with the R_1C_1 lowpass network. Hence, the active lowpass filter also acts as a buffer for the signal x(t) from the above mentioned sinusoidal signal source. The cutoff frequency of this lowpass filter can be varied by varying the value of R_1 as shown in figure 7.15.2.

The gain trim buffer amplifier shown in figure 7.15.2 is implemented by wiring the second 741 operational amplifier as a follower with gain. The gain of the circuit can be varied by varying the ratio R_3/R_2 of the resistors R_3 and R_2. Thus, the analogue input signal x(t) can be properly scaled to suit the input dynamic range of the analogue-to-digital converter for full scale conversions and also maximum signal-to-noise ratio. This is the case when the amplitude of the input signal is normalized with respect to the maximum input voltage of the analogue-to-digital converter to be described next.

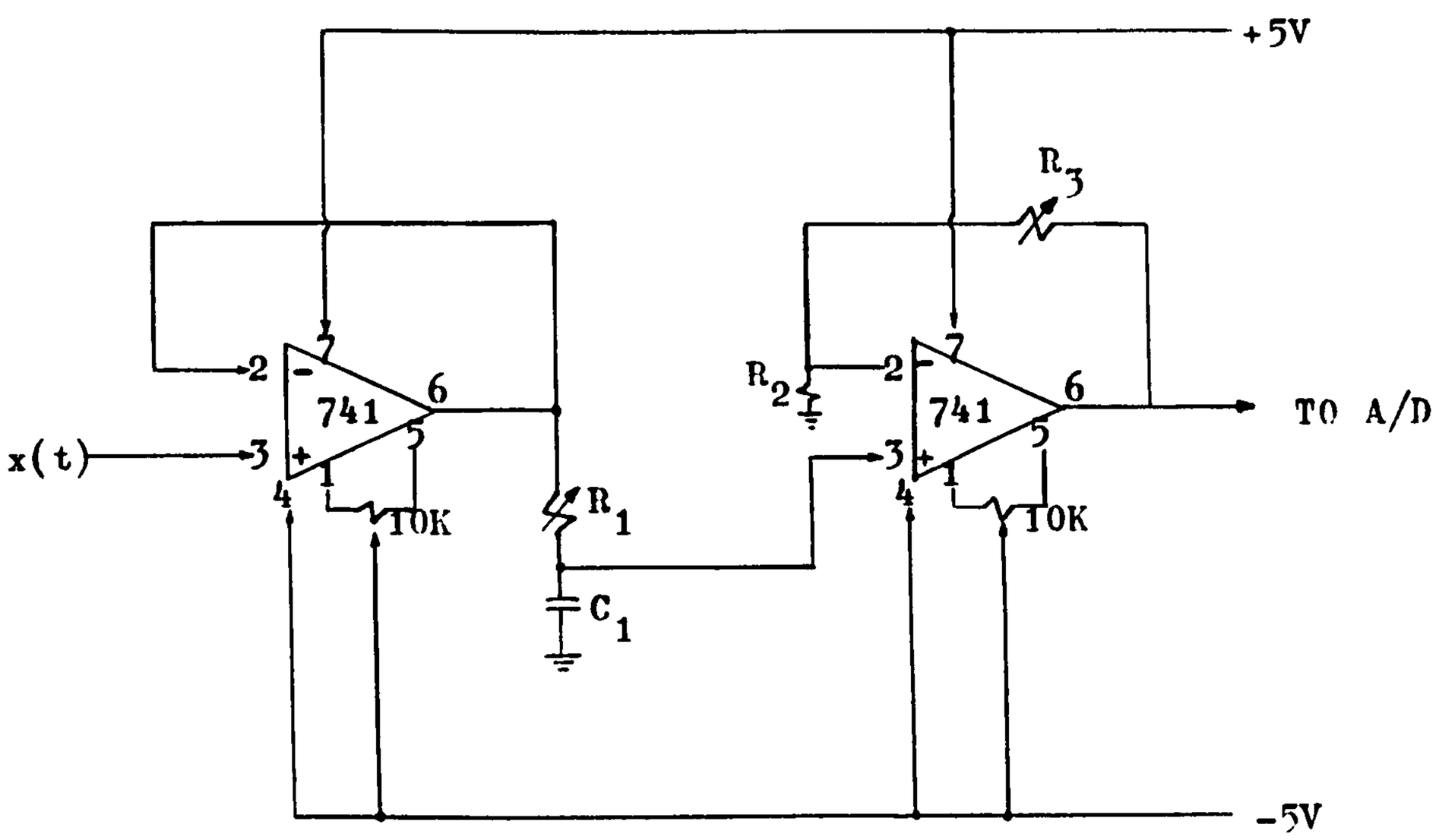


FIG: 7.15.2 CIRCUIT DIAGRAM OF THE BAND-LIMITING LOWPASS FILTER AND THE CASCADED BUFFER AMPLIFIER

7.15.3 THE BIPOLAR ANALOGUE-TO-DIGITAL INTERFACING:

The analogue-to-digital converter (A/D) used for processing analogue signals with the demonstration processor was specially designed from a low-cost digital-to-analogue (D/A) as shown in figure 7.15.3. Since it is of the counting ramp type of analogue-to-digital converter, the design shown in figure 7.15.3 is only capable of a slow

conversion rate. It was then found that for this slow conversion rate, a sample-and-hold circuit was not required in the sampling process.

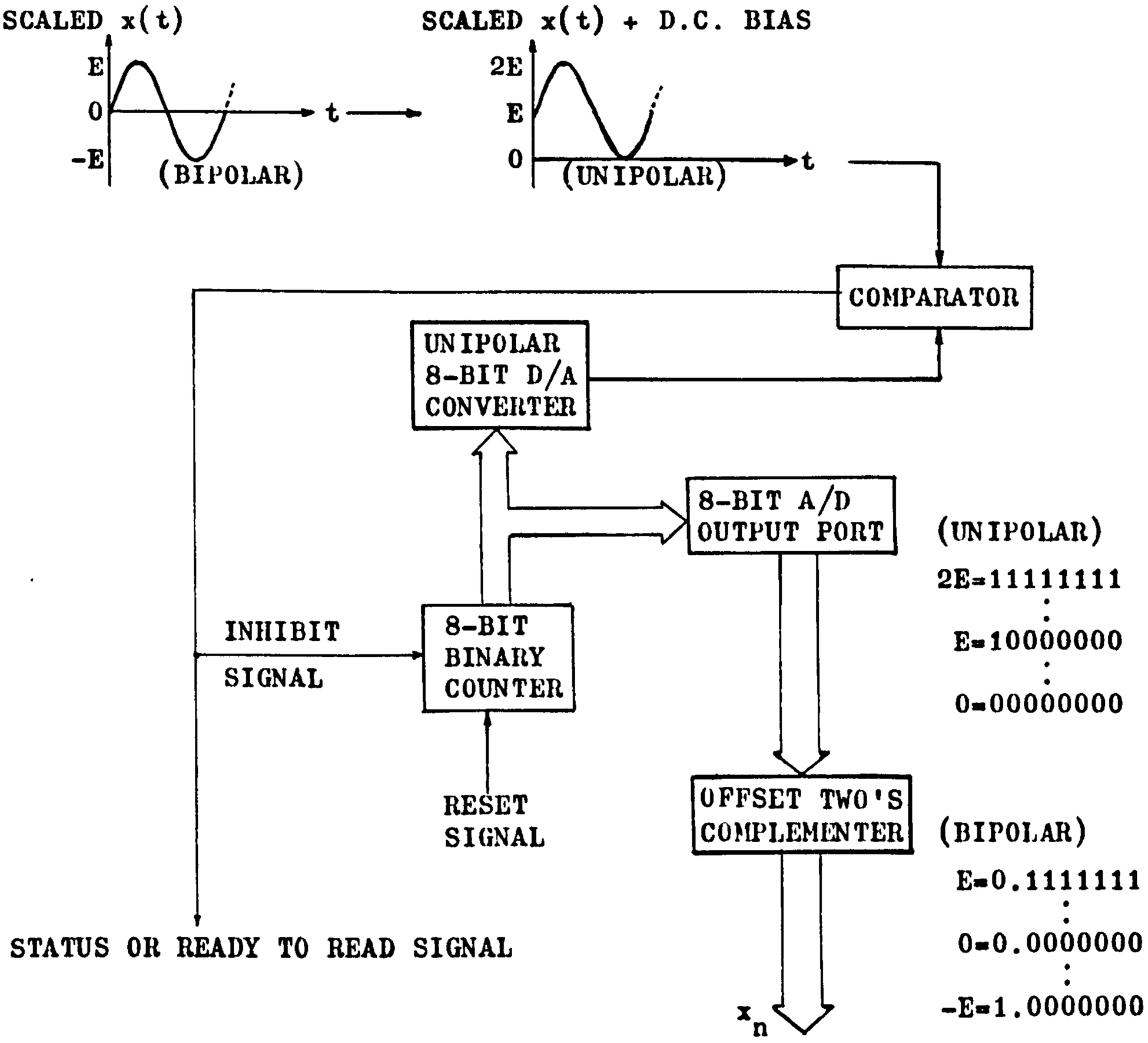


FIG: 7.15.3 BLOCK DIAGRAM OF THE BIPOLAR ANALOGUE-TO-DIGITAL CONVERTER USED FOR PROCESSING ANALOGUE SIGNALS WITH THE DEMONSTRATION PROCESSOR

The block diagram above shows the technique used to convert a unipolar digital-to-analogue converter (D/A) into a bipolar analogue-to-digital converter (A/D) whose output is in two's complement notation in order to be compatible with the input format of the demonstration processor. In the conversion process, the scaled x(t) from the output of the gain trim buffer amplifier in figure 7.15.2,

is bipolar and has a voltage swing of $\pm E$ which is, therefore, the maximum input voltage swing of the above bipolar analogue-to-digital converter (A/D). The bipolar scaled x(t) to be sampled, is first made uipolar by superimposing on it a d.c. bias E. The unipolar 8-bit digital-to-analogue converter (D/A) is used in a feedback loop to produce an analogue ramp output in the analogue-to-digital conversion process, which is then compared with the above unipolar analogue input signal x(t), previously scaled to exploit the full dynamic range of the bipolar analogue-to-digital converter (A/D). The output of the comparator therefore controls the status of the analogue-to-digital converter (A/D).

After each complete analogue-to-digital conversion, the output of the 8-bit binary counter is then loaded in parallel into the output port of the analogue-to-digital converter, and the converter is now ready for the next conversion process. The output of the output port is converted into two's complement notation via the offset two's complementer as shown in figure 7.15.3. This is then the digital input x_n which is then loaded in parallel into the input port (IP) of the demonstration processor, as previously described in section 7.3. In the above conversion process, the bipolar scaled x(t) is converted into the digital signal x_n which is in 7-bit plus sign two's complement notation. Hence, the dynamic range of the above bipolar analogue-to-digital converter (A/D) is, therefore,

$$-1 \leqslant x_n \leqslant (1-2^{-7})$$

The detail circuit diagram of this bipolar analogue-to-digital converter (A/D) is shown in figure 7.15.4 and described below. The 8-bit unipolar digital-to-analogue converter (D/A) used, was a Ferranti ZN425E chip which had also an 8-bit binary counter on chip.

In order to turn the scaled x(t) into a unipolar input signal, a d.c. bias of magnitude E is tapped out of a 5K potentiometer, with a 5V potential across it, and superimposed on the scaled x(t) as shown in figure 7.15.4. This unipolar signal is then connected to one of the two inputs of the comparator implemented by a 741 operational amplifier as shown in figure 7.15.4. The other input (V_{ref}) of the comparator is connected to the analogue output (AO) of the 8-bit digital-to-analogue converter (D/A). This analogue output (AO) is in

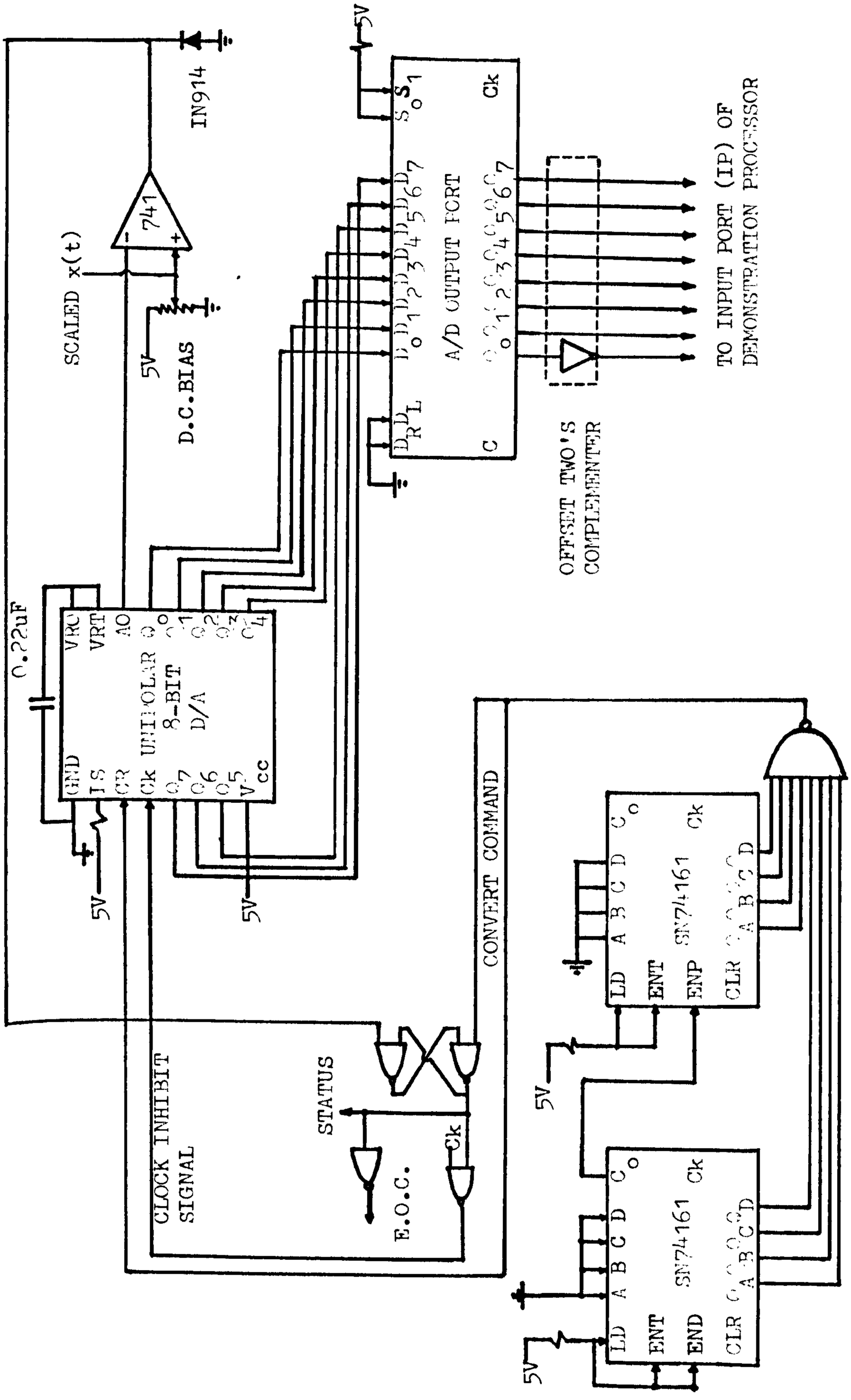


FIG: 7.15.4 LOGIC DIAGRAM OF THE BIPOLAR ANALOGUE-TO-DIGITAL CONVERTER USED FOR PROCESSING ANALOGUE SIGNALS WITH THE DEMONSTRATION PROCESSOR.

fact a ramp signal which goes up to a maximum of 256 quantization steps, and acts as a reference voltage in the comparison process. The operation of the comparator is as follows. The output of the comparator switches between two voltage levels which are made to correspond to the logical 1 and 0 voltage levels in the denonstration processor by setting the bias voltages of the 741 operational amplifier as shown in figure 7.15.4. When the magnitude of the unipolar scaled x(t) exceeds that of the ramp output mentioned above, the comparator output is then a logical 1, but is a logical 0 if otherwise. It starts to switch at the point when the above ramp signal is equal to the magnitude of the unipolar scaled x(t). The diode (IN914) is used to impose a voltage bound for the output of the comparator and to enhance its switching for slow-varying inputs. (the characteristics of the diode above should be so chosen as to give a forward-bias voltage drop of approximately 0.6 volt).

As shown in figure 7.15.4, the two 4-bit synchronous divide-by-16 counters (SN74161's) are connected as a divide-by-256 synchronous counter whose 8-bit output is decoded via the 8-input NAND gate (SN74S30) to give an output pulse every 256 clock cycles. The output of this 8-input NAND gate then furnishes a convert command for an analogue-to-digital conversion, and is connected to the counter reset input (CR) on the ZN425E chip. On the negative-going edge of the convert command pulse, the on-chip 8-bit binary counter of the ZN425E chip is reset to zero and the status signal to logical 1. On the positive-going edge of the convert command pulse, the on-chip 8-bit binary counter then starts to count up from zero. The analogue output (A0) of the 8-bit digital-to-analogue converter (D/A) ramps until it is equal to the unipolar scaled x(t) applied to the other input of the comparator. At this point, any further clock pulses are inhibited and an end-of-conversion (EOC) signal is then generated. The status signal also goes low (logical 0) to indicate that the output data is valid. This output data is held constant until 256 clock cycles have elapsed, which correspond to one full-scale conversion. Thus, every fresh output data sample will be available within 256 clock cycles which is then the cycle-time of the bipolar analogue-to-digital converter (A/D). For instance, if the clock frequency is 256KHz, the cycle-time is then 1ms. To obtain

a linearity error of $\pm$0.5 least significant bit, the conversion rate of the above bipolar analogue-to-digital converter (A/D) is just under 2KHz. The above described mode of operation of the bipolar analogue-to-digital converter (A/D) is the "clocked realtime mode" although the cycle-time of the bipolar analogue-to-digital converter (A/D) can be shortened by changing its operation into the "free-running non-realtime mode" whence the output data samples are furnished at irregular intervals, marked by the end-of-conversion (EOC) pulses. The former mode of operation was used since the demonstration processor was employed to process analogue signals in realtime. The control and timing waveforms of the above bipolar analogue-to-digital converter (A/D) are shown in figure 7.15.5 below.

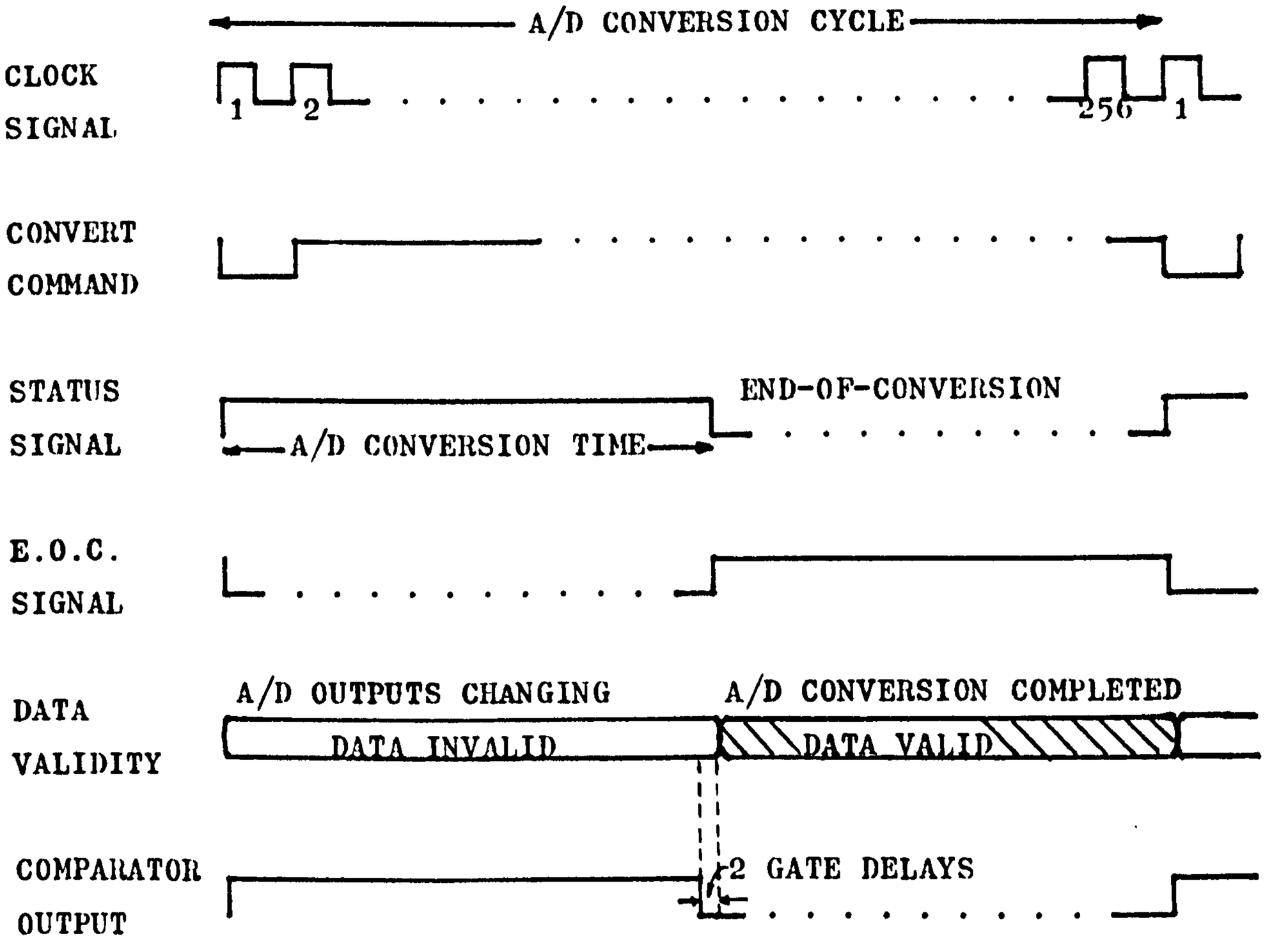


FIG: 7.15.5 CONTROL AND TIMING WAVEFORMS OF THE BIPOLAR ANALOGUE-TO-DIGITAL CONVERTER (A/D)

As shown in figure 7.15.4, the output data valid after each analogue-to-digital conversion is first clocked into the A/D output port by an end-of-conversion (EOC) pulse generated at the end of the conversion. The A/D output port is in fact a parallel-in-parallel-out 8-bit register, and was implemented with a package of SN74198. As shown in figure 7.15.3, the output of this 8-bit A/D output port is still unipolar and in straight binary format. In order to be compatible with the input format of the demonstration processor, the output of the A/D output port is then passed through the offset two's complementer to become the bipolar two's complement digital signal x_n, coded in eight bits including sign. The offset two's complementer functions by simply changing the sign bit of the 8-bit output of the A/D output port as shown in figure 7.15.3. It was implemented with an inverter gate ($\frac{1}{6}$SN74S04) connected as shown in figure 7.14.4.

As shown in figures 7.15.3 and 7.15.4, the output of the offset two's complementer is in fact the output of the bipolar analogue-to-digital converter (A/D), which is ready to be loaded in parallel into the input port (IP) of the demonstration processor. However, two factors have to be considered before a successful interfacing of the bipolar analogue-to-digital converter (A/D) with the demonstration processor can be achieved. First of all, since the bipolar analogue-to-digital converter (A/D) has a cycle-time of 256 clock cycles and the demonstration processor has a cycle-time of 9 clock cycles, a frequency translater is therefore required to translate the higher clock rate of the bipolar analogue-to-digital converter (A/D) to that of the demonstration processor. Secondly, the operation of the bipolar analogue-to-digital converter (A/D) has to be synchronized with that of the demonstration processor so that data transfers between the two devices are correctly timed.

The logic diagram of the above frequency translater used in interfacing the bipolar analogue-to-digital converter (A/D) with the demonstration processor is shown in figure 7.15.6. Since the demonstration processor requires nine clock cycles to produce an output sample, the frequency translater therefore translates the clock rate f_a of the bipolar analogue-to-digital converter (A/D) to f_d, the clock rate of the demonstration processor, where $f_d = 9f_{a/256}$, such that each input sample x_n from the bipolar analogue-to-digital

converter (A/D) can be processed synchronously by the demonstration processor.

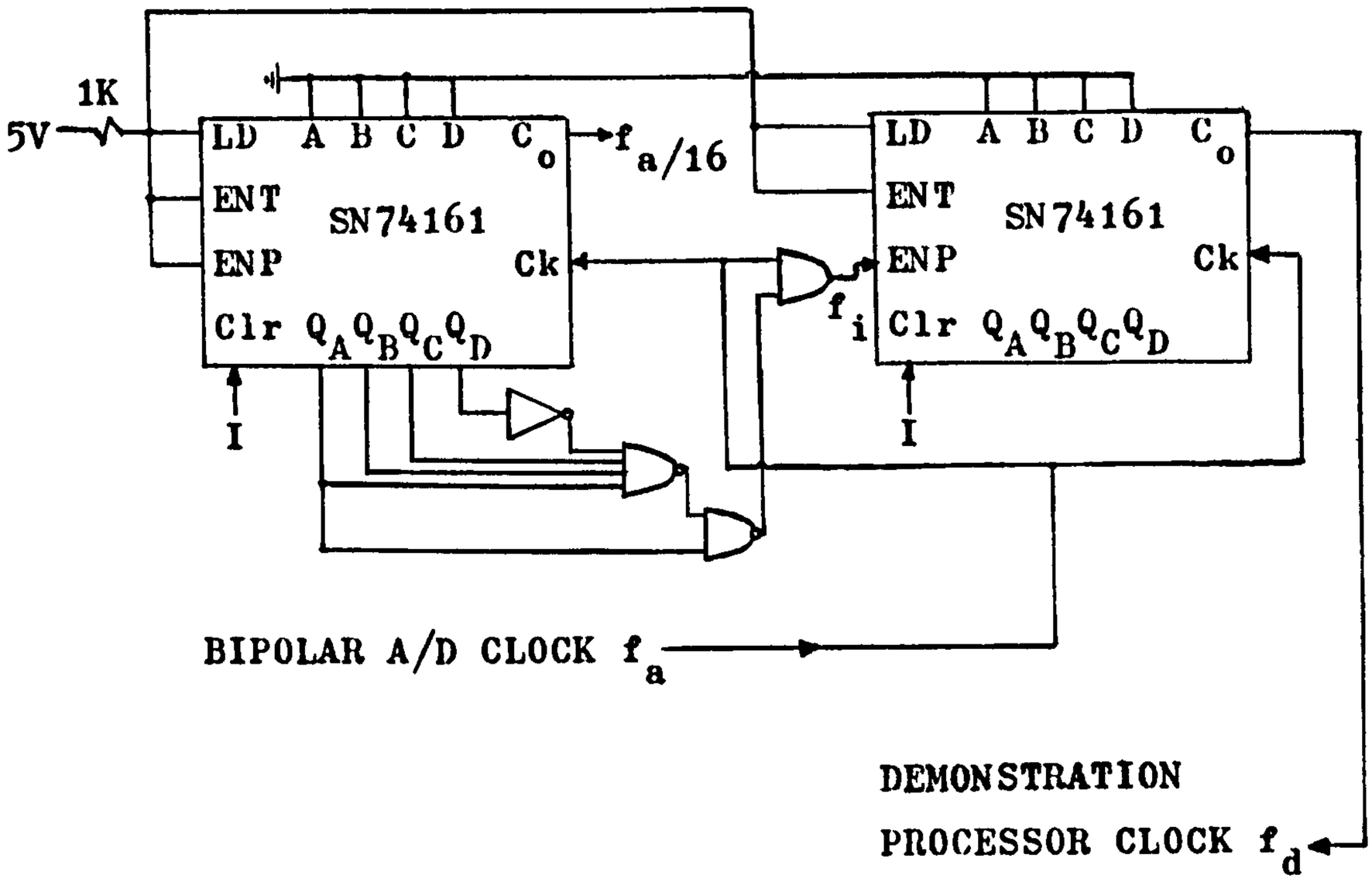


FIG: 7.15.6 LOGIC DIAGRAM OF THE FREQUENCY TRANSLATER

As shown in figure 7.15.6 above, the frequency translater is implemented by modifying the connections of the two divide-by-16 synchronous counters (SN74161's) shown in figure 7.15.4 previously. The frequency translation mentioned previously is accomplished in two stages. The first stage of translation is accomplished by sampling the outputs of the first divide-by-16 synchronous counter to yield an intermediate frequency f_i as follows.

$$f_i = 9f_{a/16}$$
$$= 0.5625\ f_a$$
$$=(0.1001)f_a$$

If the binary factor (1001) in the equation for f_i above corresponds to the outputs Q_A, Q_B, Q_C, Q_D, of the first divide-by-16 synchronous counter, then f_i can simply be obtained by selecting these outputs according to the logic equation:

$$f_i = \bar{Q}_A.Ck + Q_A.Q_B.Q_C.\bar{Q}_D.Ck$$

$$= (Q_A.\overline{Q_A.Q_B.Q_C.\bar{Q}_D}).Ck$$

Thus there are nine clock pulses in f_i, the intermediate frequency, for every sixteen clock pulses in f_a. In the second stage of translation, f_i is then passed into the second divide-by-16 counter as shown in figure 7.15.6, to give $f_d = 9f_a/256$.

Finally, synchronized data transfers between the bipolar analogue-to-digital converter (A/D) and the demonstration processor is achieved by digging a control signal from the demonstration processor to synchronize its operation with that of the bipolar analogue-to-digital converter (A/D). Furthermore, this control signal was made the convert command of the analogue-to-digital conversion process such that the above frequency translater then directly replaced the two divide-by-16 synchronous counter shown in figure 7.15.4 previously. Since a convert command pulse must fall between two end-of-conversion pulses (see figure 7.15.5), it will then be save to transfer data from the bipolar analogue-to-digital converter (A/D) to the demonstration processor during this period of time. The convert command signal was chosen to be $\overline{\phi_2.f_a}$, where ϕ_2 is the control signal derived from the control and timing unit of the demonstration processor as described in section 7.2 previously. The clock input to the demonstration processor is now f_d while that of the bipolar analogue-to-digital converter (A/D) is f_a.

7.15.4 THE BIPOLAR DIGITAL-TO-ANALOGUE INTERFACING:

Like the bipolar analogue-to-digital converter (A/D), the bipolar digital-to-analogue converter (D/A) was designed with the Ferranti ZN425E chip as shown in figure 7.15.7. It is interfaced with the $SR_{y_{n-2}}$ in the output port (OP) of the demonstration processor. (when the demonstration processor is being multiplexed to implement a higher order filter, say an eighth-order bandpass filter, the above bipolar digital-to-analogue converter (D/A) will then be interfaced with the register containing the output Y_{n-2} of the bandpass filter shown in figure 7.13.2). Also, as in the case of the bipolar analogue-to-digital converter (A/D), in changing an 8-bit

unipolar digital-to-analogue converter (D/A) to a bipolar 7-bit plus sign two's complement digital-to-analogue converter (D/A), one bit is lost in the dynamic range of the original 8-bit unipolar digital-to-analogue converter (D/A). Hence, the input dynamic range of this bipolar digital-to-analogue converter (D/A) is given by

$$-1 \leqslant y_{n-2} \leqslant (1-2^{-7})$$

As shown in figure 7.15.7, the bipolar two's complement signal y_{n-2} from the output port (OP) of the demonstration processor, is first converted to a unipolar straight binary signal so that it can be accommodeated in the unipolar 8-bit digital-to-analogue converter (D/A). This is performed by the offset two's complementer which changes the sign of each y_{n-2} sample loaded in parallel into the input port of the bipolar digital-to-analogue converter (D/A). The offset two's complementer was implemented with one inverter gate ($\frac{1}{6}$SN74S04). The input port of the bipolar digital-to-analogue converter (D/A) is in fact a parallel-in-parallel-out register and was implemented with one package of SN74198. This input port is clocked by the control signal $\emptyset_4$ obtained from the control and timing unit of the demonstration processor described previously. The output of this input port is connected, via the offset two's complementer, to the 8-bit unipolar digital-to-analogue converter (D/A). (when the logic select (LS) input on the ZN425E chip is taken low, its Q's outputs become data inputs). Furthermore, the data inputs of the above input port are connected to the eight most significant bits of the shift register $SR_{y_{n-2}}$ in the output port (OP) of the demonstration processor. As data is right-shifted into the shift register $SR_{y_{n-2}}$ in the first eight clock cycles of the operation of the processor, there is no data transfers between the demonstration processor and the bipolar digital-to-analogue converter (D/A). At the start of the 9th clock cycle, the data in the shift register $SR_{y_{n-2}}$ is then loaded in parallel into the input port of the bipolar digital-to-analogue converter (D/A) for conversion to an output analogue signal. This output signal is taken from the analogue output (AO) pin on the ZN425E chip, which is further passed on to the reconstruction lowpass filter and the analogue output buffer to be discussed in the next section.

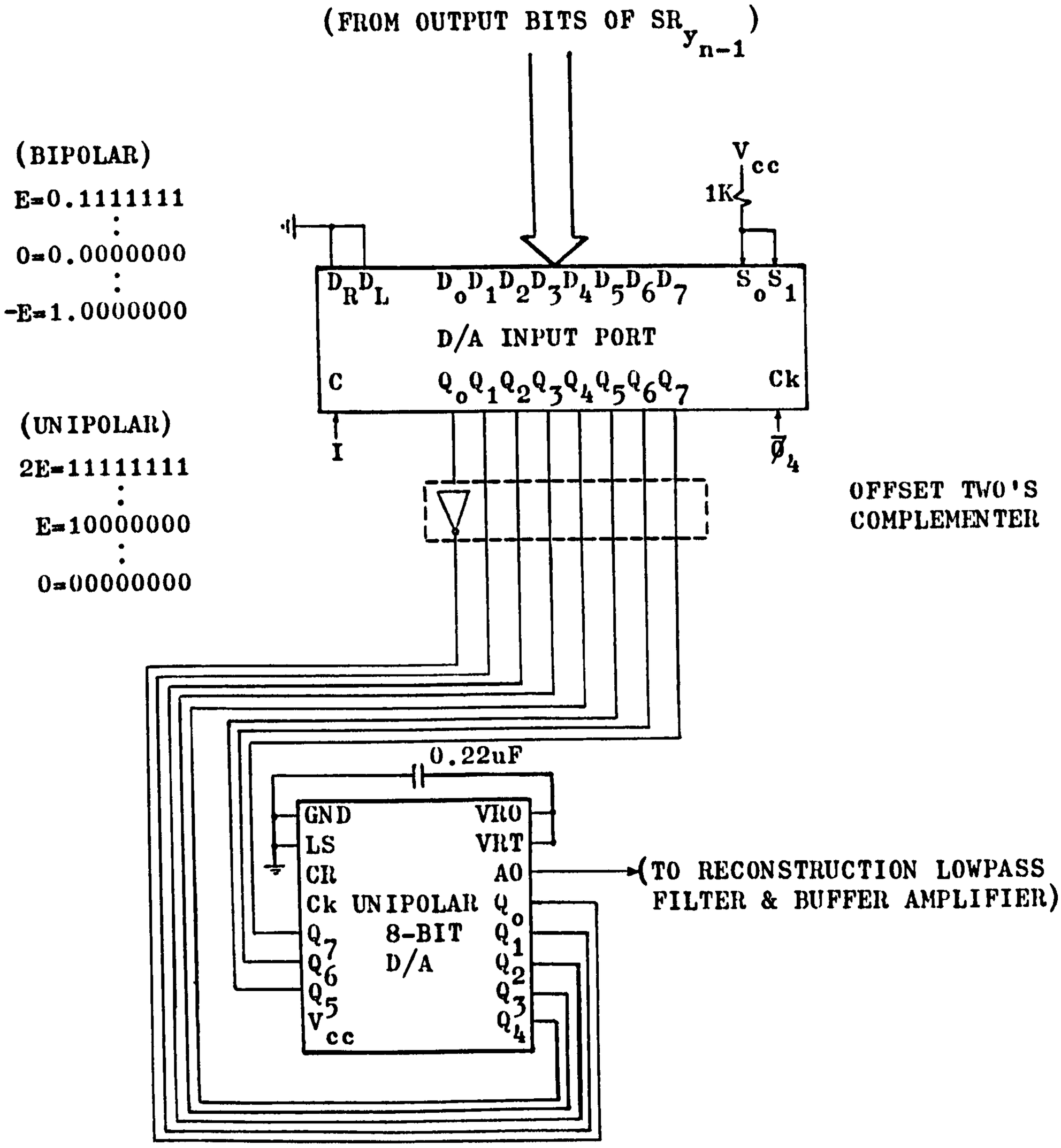


FIG: 7.15.7 LOGIC DIAGRAM OF THE BIPOLAR DIGITAL-TO-ANALOGUE CONVERTER USED FOR PROCESSING ANALOGUE SIGNALS WITH THE DEMONSTRATION PROCESSOR

7.15.5 THE RECONSTRUCTION LOWPASS FILTER AND BUFFER AMPLIFIER:

As shown in figure 7.15.1, the output of the above bipolar digital-to-analogue converter (D/A) has to be passed through an output reconstruction lowpass filter in order to filter out higher harmonics present due to the sampling process. Furthermore, due

the sign change introduced by the offset two's complementer in the bipolar digital-to-analogue converter (D/A), there is a positive d.c. bias of E=2.55V present at the analogue output (A0). In order to provide a bipolar analogue output from the bipolar digital-to-analogue converter, the output buffer amplifier was designed with an offset characteristic as shown in figure 7.15.8 below to remove the above mentioned positive d.c. bias of 2.55V.

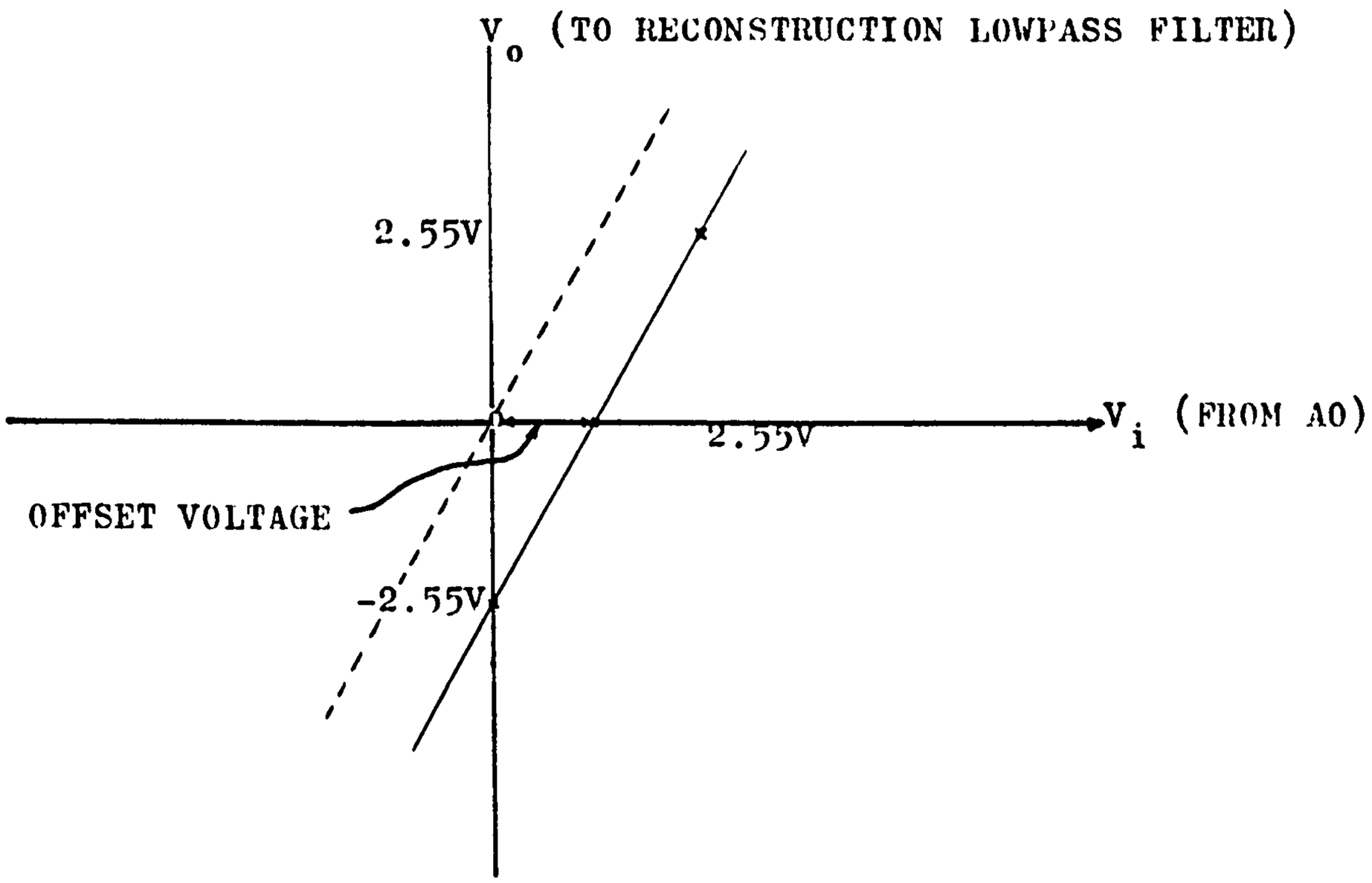


FIG: 7.15.8 TRANSFER CHARACTERISTIC OF THE OFFSET OUTPUT BUFFER AMPLIFIER

Figure 7.15.9 shows the circuitry of the above offset output buffer amplifier and the output reconstruction lowpass filter. The analogue output buffer amplifier is made up of two 741 operational amplifiers. The first amplifier buffers the analogue output (A0) of the bipolar digital-to-analogue converter (D/A) and also provides the above mentioned offset characteristic, while the second amplifier is combined with the RC network to form an active lowpass filter.

The analogue output (A0) of the bipolar digital-to-analogue converter (D/A) is connected to the positive input of the first 741

operational amplifier whose output is being fed back to its negative input via the resistive network shown. The trimmer pot P_1 allows the amplifier output to be set to -2.55V when the input analogue signal from the bipolar digital-to-analogue converter (D/A) is zero (i.e when y_{n-2} = 1.0000000). The amplifier output is then set to 2.55V by the trimmer pot P_2 when the output analogue signal from the bipolar digital-to-analogue converter (D/A) is a maximum of 2.55V (i.e. when y_{n-2} = 0.1111111). Provided that these trimmer pots are tuned in the order given, that is tune P_2 after P_1, they do not interect with each other.

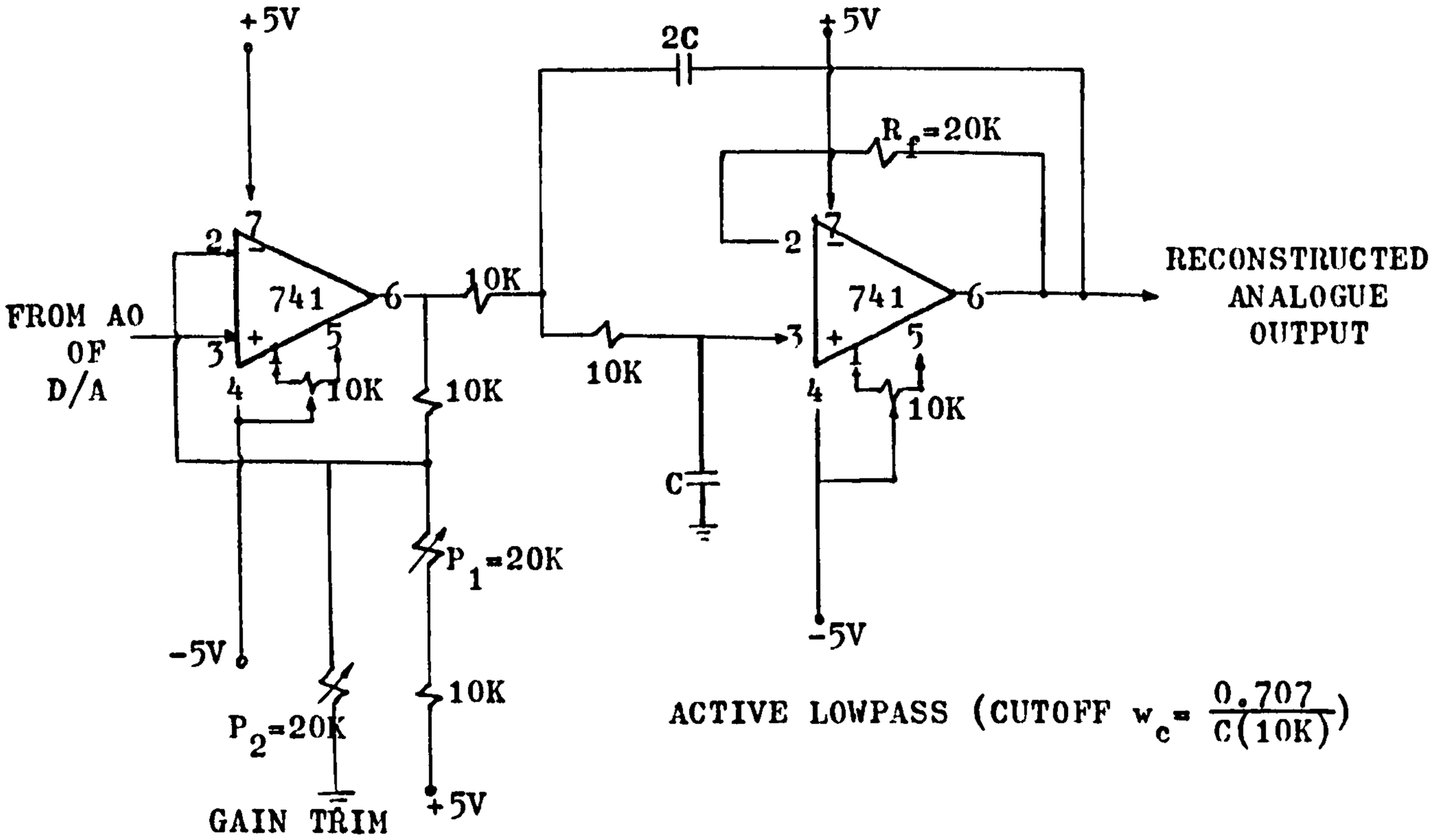


FIG: 7.15.9 CIRCUIT DIAGRAM OF THE RECONSTRUCTION LOWPASS FILTER AND THE OUTPUT OFFSET BUFFER AMPLIFIER

The above active reconstruction lowpass produces a roll-off of -40dB/decade. The cutoff frequency w_c is given by the expression in the bracket above, and the filter is seen to be of the Butterworth type. The second 741 operational amplifier is connected for d.c unity gain while the feedback resistor R_f is included for d.c. offset. The cutoff frequency w_c can be made variable by making C variable.

7.16 CONCLUSION:

In this chapter, we have presented the logic design and construction of the various constituent units of the demonstration processor purposely built to verify experimentally the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) discussed in chapter five, and to investigate the proposed generalized noise model for the finite word length effects on the hardware implementation of the above CBFCS algorithm described in chapter six. In order to process analogue signals, the above demonstration processor was then interfaced to its peripheral units viz., the bipolar analogue-to-digital converter (A/D) and the bipolar digital-to-analogue converter (D/A). The logic design of the above peripheral units and their respective interfacing with the demonstration processor has also been presented.

Experimental results of some bandpass filters designed by the proposed design technique discussed in chapter three will be presented in the next chapter. In obtaining the impulse responses of these bandpass filters, a digital impulse was then loaded into the input port (IP) of the demonstration processor which was thereafter stepped through the cycles of these impulse responses whose digital values were then read off at the output port (OP) of the processor. On the other hand, in obtaining frequency responses to analogue input signals the peripheral units of the demonstration processor were also used. For this purpose, a sinusoidal signal generator was connected to the bipolar analogue-to-digital converter (A/D) so that frequency spot checks could then be made and the filtered analogue outputs obtained via the bipolar digital-to-analogue converter (D/A). However, owing to the slow conversion rate of the above bipolar analogue-to-digital converter (A/D) as described in the last section, experimental results of the frequency responses of a bandpass filter, centred at 120Hz with a sampling frequency of 1.2KHz, and its constituent elemental filters were only obtained. The experimental results presented in the next chapter will also be analysed along with some relevant simulation results to be presented in the next chapter as well.

CHAPTER EIGHT

RESULTS, ANALYSES AND DISCUSSIONS

8.1 INTRODUCTION:

As mentioned previously in chapter seven, the demonstration processor and its associated peripheral units described there were built to verify experimentally the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) discussed in chapter five, and to investigate the proposed generalized quantization noise model discussed in chapter six for the error analysis of the finite word length effects on the hardware implementation of the proposed scheme above. Furthermore, the word length of the demonstration processor was chosen to be sufficiently short to investigate the adverse effects of such a short word length on the proposed design technique discussed in chapter three. In addition to experimental results obtained via the demonstration processor, computer simulation results were also obtained for longer word lengths, using a CDC 6400 general-purpose computer, in the course of the investigation of the above mentioned proposed design technique.

In this chapter, we shall present and analyse the above results. Broadly speaking, the present chapter can be divided into two parts. The former part deals with the above results in the time domain. This includes some impulse responses for two configurations of realizing the proposed design technique discussed in chapter three. Limit cycle behaviours of these impulse responses have also been analysed. The latter part deals with the results obtained in the frequency domain, that is, frequency responses. The effect of finite word length has also been subsequently analysed. Additionally, relevant observations have been made and discussed while conclusions have been drawn from the analyses of the above results.

Figure 8.1.1 shows the experimental set-up used in the experimental investigation of the above mentioned proposed design technique. As mentioned previously in the conclusion of the last chapter, experimental digital impulse responses were obtained by applying a digital impulse (via some manual switches simulating the binary bits) to the filter under test. On the other hand, experimental

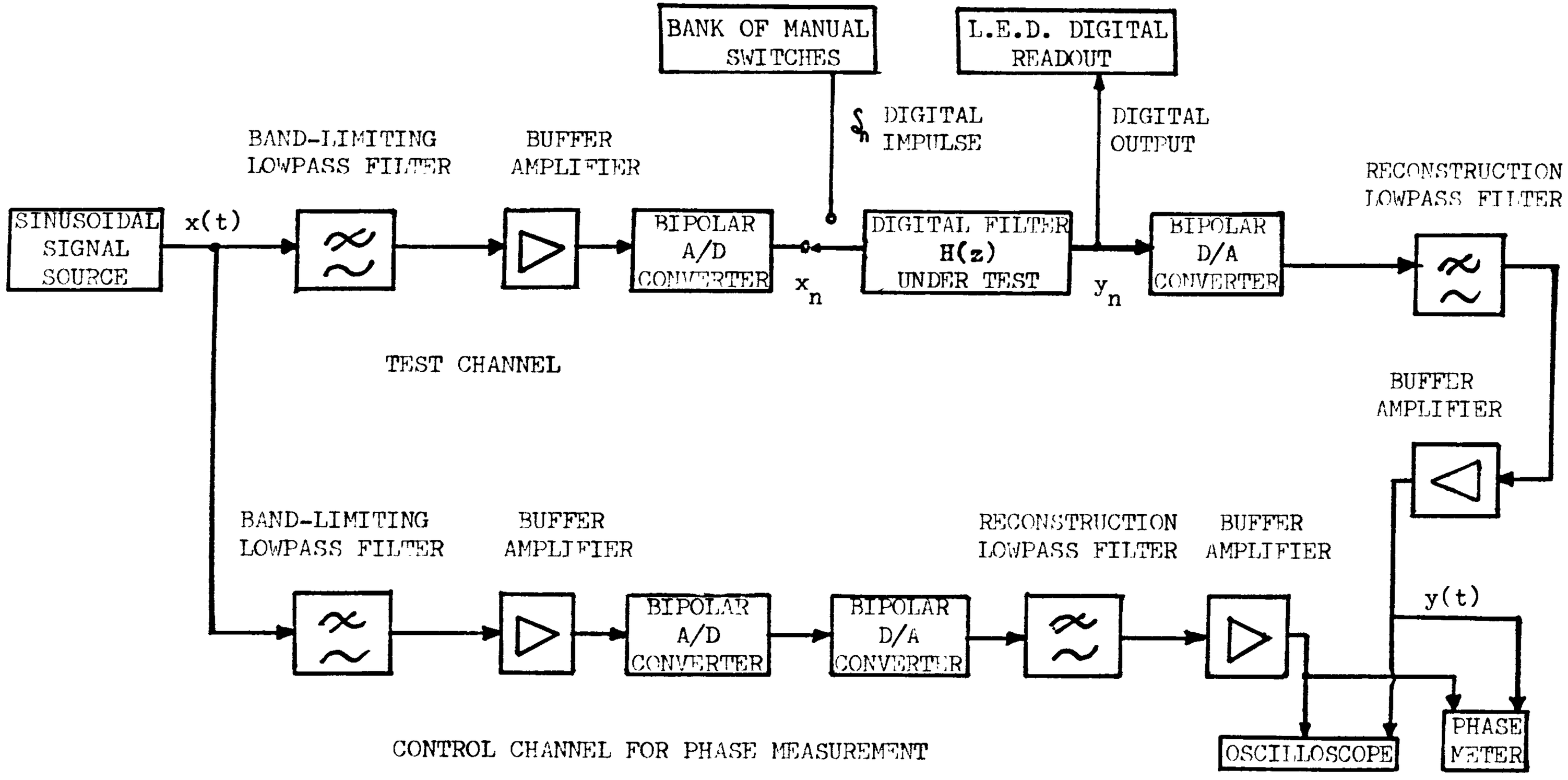


FIG: 8.1.1 EXPERIMENTAL SET-UP USED FOR FREQUENCY, PHASE AND IMPULSE RESPONSES MEASUREMENT. CONTROL CHANNEI SHOWN ABOVE CONSISTS OF LOWPASS FILTERS, BUFFER AMPLIFIERS, A/D CONVERTER AND D/A CONVERTER IDENTICAL TO THOSE USED IN THE TEST CHANNEL.

frequency responses were obtained by making frequency spot checks with analogue sinusoidal inputs applied to the filter under test via the peripheral units.

The filters tested were first designed by the proposed design technique discussed in chapter three, according to the general design model described in section 3.2.4 and shown in figure 3.2.8. The resultant equation for the transfer function H(z), which was then quantized via the quantized procedure described in section 6.4.2 to yield the quantized transfer function $H_Q(z)$, is given as follows:

$$H(z) = \frac{W_n(1 - z^{-1}.e^{-aT}\cos((\omega_p - b)T))}{1 - z^{-1}.2e^{-aT}\cos((\omega_p - b)T) + z^{-2}.e^{-2aT}} -$$

$$\frac{(1 - z^{-1}.e^{-aT}\cos(\omega_p T))}{1 - z^{-1}.2e^{-aT}\cos(\omega_p T) + z^{-2}.e^{-2aT}} +$$

$$\frac{(1 - z^{-1}.e^{-aT}\cos(\omega_p T))}{1 - z^{-1}.2e^{-aT}\cos(\omega_p T) + z^{-2}.e^{-2aT}} -$$

$$\frac{W_n(1 - z^{-1}.e^{-aT}\cos((\omega_p + b)T))}{1 - z^{-1}.2e^{-aT}\cos((\omega_p + b)T) + z^{-2}.e^{-2aT}} \qquad (8.1.1)$$

where $\omega_p - \omega_p = 2b$, and all symbols have their usual meanings as given and described in section 3.2 in chapter three. Furthermore, as explained in section 3.5, and by equation (3.5.6), equation (8.1.1) can be rewritten as:

$$H(z) = H_1^T(z) - H_1^P(z) + H_2^P(z) - H_u^T(z) \qquad (8.1.2)$$

where the first $H_1^T(z)$ is the transfer function for the lower transition pole and the fourth $H_u^T(z)$ is the transfer function for the upper transition pole. $H_1^P(z)$ and $H_2^P(z)$ are the transfer functions for the passband poles.

For the realization of equation (8.1.1), two configurations are possible as shown in figure 8.1.2 below:

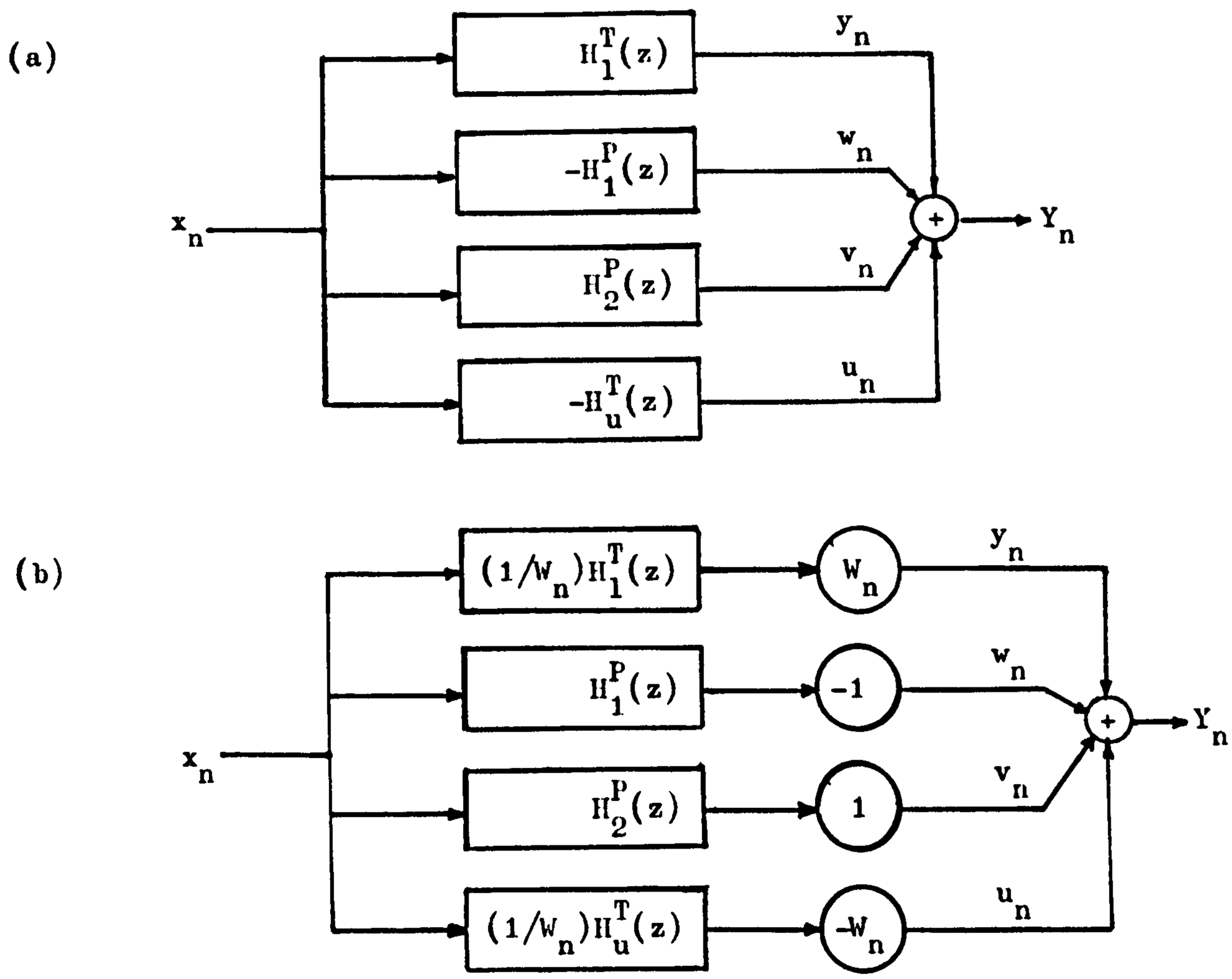


FIG: 8.1.2 TWO POSSIBLE CONFIGURATIONS FOR THE REALIZATION OF AN 8TH-ORDER BANDPASS FILTER.

The above two configurations will be investigated in this chapter. Their respective models for computer simulations are shown in figures 8.1.3 and 8.1.4. In addition, the coefficients shown in the above figures were quantized in all simulations done. The above mentioned models for simulations are derived from the generalized quantization noise model discussed and described in section 6.6 in chapter six.

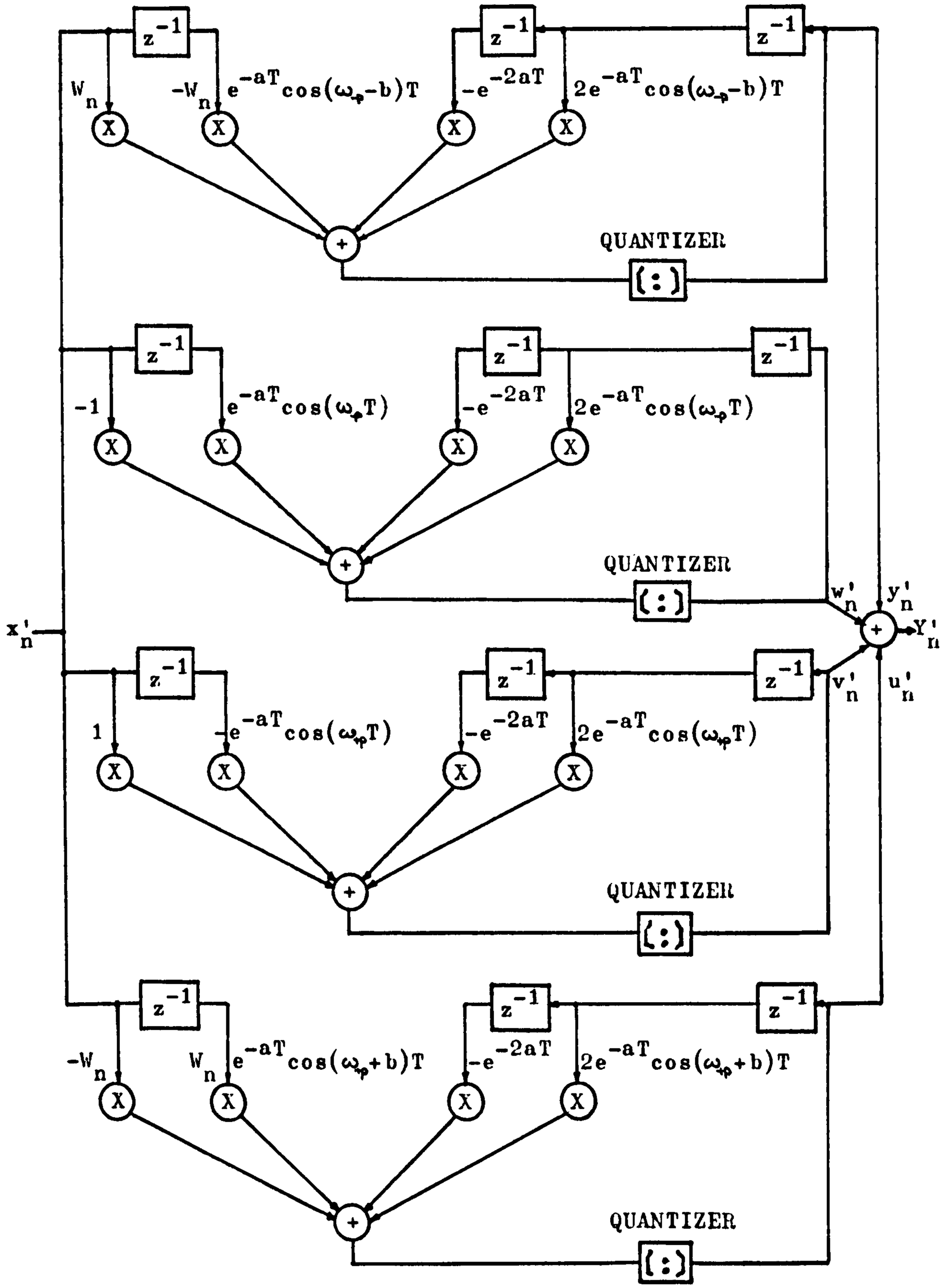


FIG: 8.1.3 SIMULATION MODEL FOR THE REALIZATION CONFIGURATION IN FIGURE 8.1.2(a)

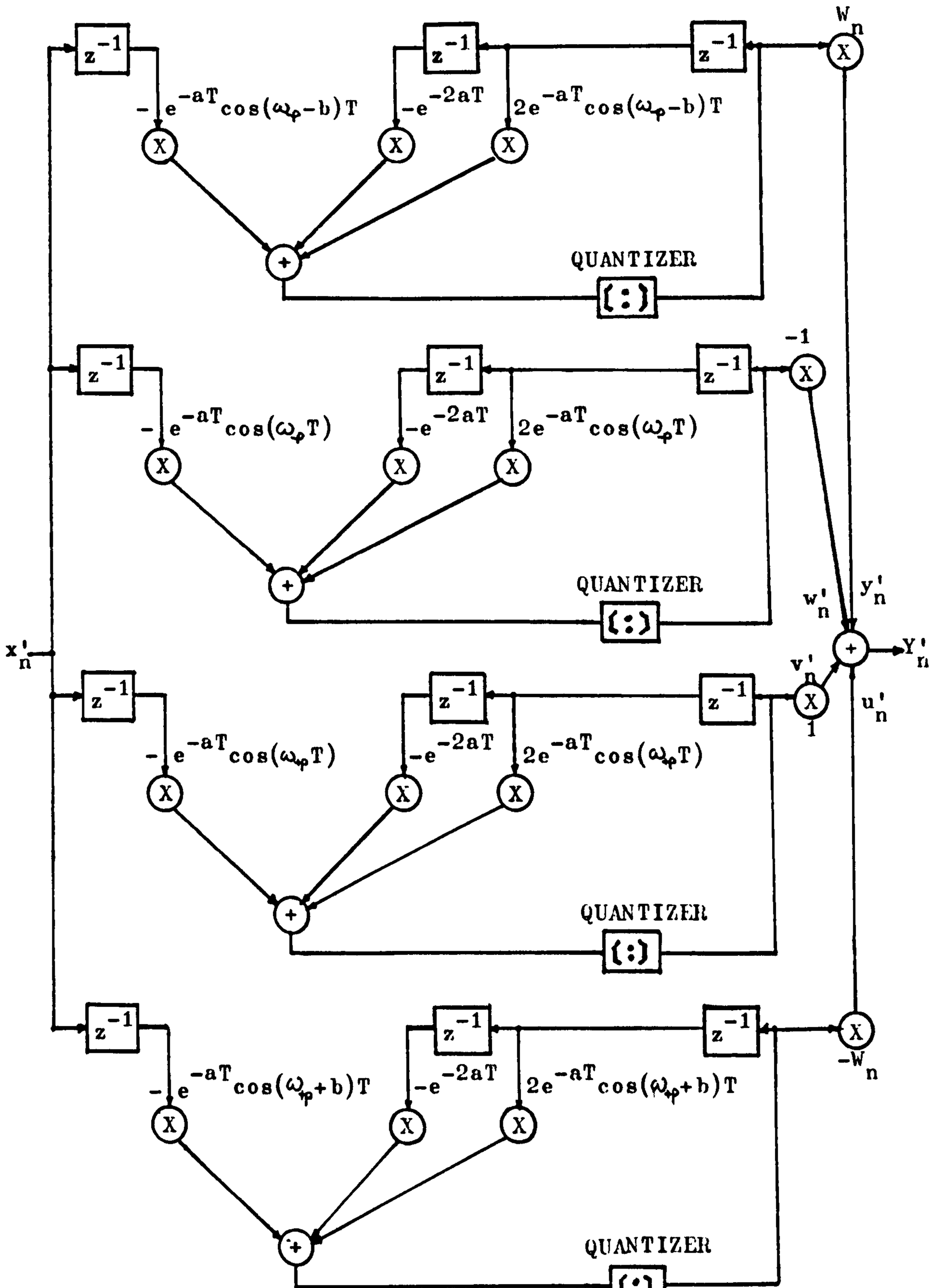


FIG: 8.1.4 SIMULATION MODEL FOR THE REALIZATION CONFIGURATION IN FIGURE 8.1.2(b)

8.2 TIME DOMAIN RESPONSES:

In this section, the experimental impulse responses of some bandpass filters designed by the proposed design technique are first presented for the two configurations shown in figure 8.1.2, the choice of which is then subsequently discussed. (It was possible to obtain point-by-point agreement between the above experimental impulse responses and those simulated according to figures 8.1.3 and 8.1.4.). Simulated impulse responses of the same bandpass filters, but with longer word lengths, are next presented. This is followed by an analysis of the limit cycle behaviours of the above filters, according to the generalized quantization noise model in chapter six. Furthermore, experimental results of a high-frequency bandpass filter are also presented along with some simulation results which further illustrate the effects of a finite dynamic range. Next, experimental results of a second-order section implemented seperately with the proposed "Selective Rounding Scheme discussed in chapter six and the usual two's complement up-rounding scheme used so far, are compared and discussed. Additionally, comments and relevant conclusion are given wherever appropriate.

8.2.1 EXPERIMENTAL IMPULSE RESPONSES OF SOME BANDPASS FILTERS:

In chapter three, section 3.7, the choice of sampling frequency in relation to aliasing effects was discussed. It is further seen from the index, BT, defined there as the ratio of the 3dB bandwidth of an elemental filter to the underlying sampling frequency, that unless the bandwidths of individual elemental filters used in low-frequency designs are very narrow, low-frequency filters are, in general, more affected by aliasing effects. In addition, it was also pointed out there in section 3.7, that, in some cases, an increase in the sampling frequency will help to reduce the above aliasing effects. However, there is a certain limit in practice, when the quantization effects of a finite-word-length processor is taken into account.

In view of the above, several low-frequency designs were tested and their experimental impulse responses are presented in this section while their limit cycle behaviours are discussed and analyzed in

particular. Consider next a design with the following desired specifications:

(a) A bandpass filter with a nominal bandwidth of 10Hz.

(b) Centre frequency at 120Hz, i.e., percentage bandwidth is $8\frac{1}{3}\%$.

(c) Passband ripple less than 0.2dB.

(d) The transition bands on both sides of the passband should be as narrow as possible, that is, a steep transition skirt.

In order to meet the above specifications, an eighth-order filter was used, with two passband poles and a pair of transition poles, one on each side of the passband. Furthermore, this 4-pole bandpass filter was realized with four second-order elemental filters centred at 115Hz, 117.5Hz, 122.5Hz and 125Hz respectively to give a nominal bandwidth of 10Hz. Thus, the passband poles are seperated by 5Hz in frequency while the transition poles are each seperated from its nearest passband pole by 2.5Hz in frequency. This then gave the value of b in equation (8.1.1) to be 5π radians. With this value of b, the value of a in equation (8.1.1) was further chosen to be 5π radians for large out-of-band rejection and steep transition. Hence, the 3dB bandwidth of individual elemental filters is given to be 5Hz by equation (3.7.1), and the percentage bandwidth of an elemental filter is therefore $4\frac{1}{6}\%$. However, by the discussion in section 3.7, such a percentage bandwidth of an elemental filter as given above, together with the desired specification (b), will imply a fair amount of aliasing effects at low sampling frequencies. In fact, this was found to be the case when the sampling was as low as three times the centre frequency of the bandpass filter, when aliasing effects were apparent. Consequently, the underlying sampling frequency was chosen to be 1.2kHz which further served the purpose of demonstrating the relationship of finite word length effects with an increased sampling frequency (this will become clear in later sections). This gives the ratio BT as 1:240 so that frequency aliasing effects were not apparent in the resultant design.

The resultant transfer function was then optimized according to the optimization procedure discussed in section 3.2.6 to yield the steepest transition skirt possible. The theoretical magnitude of the passband ripple before optimization is given by equation (3.2.21)

as 0.38dB approximately. However, after optimization of the transition samples, the value of the passband ripple was found to be 0.2dB approximately. The value of the optimized transition sample W_n was found to be equal to 0.499. Substituting the above values into equation (8.1.1), the resultant optimized transfer function H(z) is written as:

$$H(z) = \frac{0.499(1 - 0.8134087012z^{-1})}{1 - 1.626817402z^{-1} + 0.9741597848z^{-2}}$$

$$- \frac{(1 - 0.8060214024z^{-1})}{1 - 1.612042805z^{-1} + 0.9741597848z^{-2}}$$

$$+ \frac{(1 - 0.7908337726z^{-1})}{1 - 1.581667545z^{-1} + 0.9741597848z^{-2}}$$

$$- \frac{0.499(1 - 0.7830360431z^{-1})}{1 - 1.566072086z^{-1} + 0.9741597848z^{-2}} \qquad (8.2.1)$$

which was finally quantized as described in section 6.4.2 to give $H_Q(z)$ with coefficients represented to 16-bit accuracy including sign. The quantized transfer function is therefore written as:

$$H_Q(z) = \frac{0.5(1 - 0.8134155805z^{-1})}{1 - 1.626831161z^{-1} + 0.9741515374z^{-2}}$$

$$- \frac{(1 - 0.8060303048z^{-1})}{1 - 1.61206061z^{-1} + 0.9741515374z^{-2}}$$

$$+ \frac{(1 - 0.790832529z^{-1})}{1 - 1.581665058z^{-1} + 0.9741515374z^{-2}}$$

$$- \frac{0.5(1 - 0.783050555z^{-1})}{1 - 1.56610111z^{-1} + 0.9741515374z^{-2}} \qquad (8.2.2)$$

Note that the value of W_n has been quantized to 0.5.

Now, the quantized transfer function $H_Q(z)$ in equation (8.2.2) can be realized in two different configurations as previously shown in figure 8.1.2 in section 8.1. Configuration (b) in figure 8.1.2 was first investigated and the experimental impulse responses of the four second-order elemental filters and that of the resultant eighth-order bandpass filter are shown in figures 8.2.1 to 8.2.5. For the purpose of reference, we shall refer to the above eighth-order bandpass filter realized in configuration (b) in figure 8.1.2, with optimized transition sample W_n quantized to a value of 0.5, as bandpass filter A. The value of the unit input impulse was 0.1111111 = 0.9921875, while the respective contents of the CBFCS memories for the four second-order elemental filters are shown in tables C1 to C4 in appendix C.

Figure 8.2.1 shows the experimental impulse response $h_{n,1}$ of the second-order lower transition elemental filter $H_1^T(z)$ of the above bandpass filter A. The scaling by the quantized value of W_n was performed by one bit hard-wired right-shift of each output sample of the second-order section, since the quantized value of W_n is 0.5. It is noticed from figure 8.2.1 that the impulse response $h_{n,1}$ dies down to a limit cycle at the 187th sample. The limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is five quantization steps where the width of each quantization step is 0.0078125 since the dynamic range of the demonstration processor is eight bits including sign. Thus, the ratio of the maximum amplitude of this limit cycle to the dynamic range is 5/128=0.0390625, which is just under 4%. It is further noticed that the limit cycle is approximately sinusoidal in nature. Had there been no finite word length effects (i.e. if the elemental filter had been realized with infinite precision), the impulse response would have died down to zero value eventually without any limit cycle behaviour, as discussed in appendix B.

Figure 8.2.2 shows the experimental impulse response $h_{n,2}$ of the first second-order passband elemental filter $-H_1^P(z)$ of the above bandpass filter A. Since the output scaling factor is -1, no actual multiplier is required, and the minus sign was effected by an output subtraction. It is seen from figure 8.2.2 that the impulse response $h_{n,2}$ dies down to a limit cycle at the 162nd sample. The limit cycle

has a period L=10 so that it repeats every ten samples. Furthermore, this limit cycle has a maximum amplitude of fourteen quantization steps, such that the ratio of its maximum amplitude to the dynamic range is therefore 14/128=0.109375 or nearly 11%. It will be seen later that this is the largest of all four limit cycles of the four elemental filters that constituted the above bandpass filter A. In fact, its presence has contributed largely to the resultant amplitude of the limit cycle of the above eighth-order bandpass filter A. It is also noticed that this limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour as discussed in appendix B.

Figure 8.2.3 shows the experimental impulse response $h_{n,3}$ of the second second-order passband elemental filter $H_2^P(z)$ of the above bandpass filter A. Since the output scaling factor is unity, no multiplication is required. It is seen from figure 8.2.3 that the impulse response $h_{n,3}$ dies down to a limit cycle at the 204th sample. The limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is six quantization steps, such that the ratio of its maximum amplitude to the dynamic range is therefore 6/128=0.046875 or just about $4\frac{1}{2}$%. It is also noticed that this limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour as discussed in appendix B.

Figure 8.2.4 shows the experimental impulse response $h_{n,4}$ of the second-order upper transition elemental filter $-H_u^T(z)$ of the above bandpass filter A. The scaling by the quantized value of W_n was performed by one bit hard-wired right-shift of each output sample of the second-order section, since the quantized value of W_n is 0.5. Furthermore, the minus sign preceding $-H_u^T(z)$ was simply effected by a subtraction process. It is noticed from figure 8.2.4 that the impulse response $h_{n,4}$ dies down to a limit cycle at the 230th sample. The limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is three quantization steps, such that the ratio of its maximum amplitude to the dynamic range is therefore 3/128=0.0234375 or just under $2\frac{1}{2}$%.

It is also noticed that this limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour as discussed in appendix B.

Figure 8.2.5 shows the experimental impulse response h_n of the eighth-order bandpass filter A realized in configuration (b) shown in figure 8.1.2 with the optimized transition sample W_n quantized to a value of 0.5 and a wordlength of eight bits including sign. It is seen from figures 8.2.1 to 8.2.5 that h_n is the summation of $h_{n,1}$, $h_{n,2}$, $h_{n,3}$ and $h_{n,4}$. From figure 8.2.5, the impulse response h_n is seen to die down to a limit cycle behaviour at the 231st sample, with a period L=10, such that it repeats every ten samples. In fact, the period of the above limit cycle of bandpass filter A is the least common multiple (LCM) of the periods of the limit cycles of the individual elemental filters which constituted it. It is as expected, since the four limit cycles of the above four elemental filters are seen to beat together to form the resultant limit cycle of the above bandpass filter A. Furthermore, since individual impulse responses of the above four elemental filters are relatively displaced in time (i.e. a phase difference between adjacent pairs), certain amount of limit cycle cancellation has been observed. This is also as expected, since in equation (8.1.2) the resonance frequencies of individual elemental filters are slightly displaced in frequency as well, and alternate outputs of the above second-order elemental filters are, in fact, subtracted due to the negative weighting of their output scaling factor (see figure 8.1.2). The maximum amplitude of the limit cycle of the above bandpass filter A is sixteen quantization steps, such that the ratio of its maximum amplitude to the dynamic range is 16/128 =0.125 or $12\frac{1}{2}\%$. The magnitude of such a limit cycle is largely due to the magnitude of the limit cycle of $-H_1^P(z)$ which amounts to fourteen quantization steps alone. Had it not been for the certain amount of limit cycle cancellation mentioned above, the resultant magnitude of the limit cycle of bandpass filter A would have been larger. Again, had there been no finite word length effects, the impulse response h_n would have died down to zero eventually without any limit cycle behaviour (see a later section). Finally, the nature of the limit cycle behaviour of bandpass filter A is approximately sinusoidal.

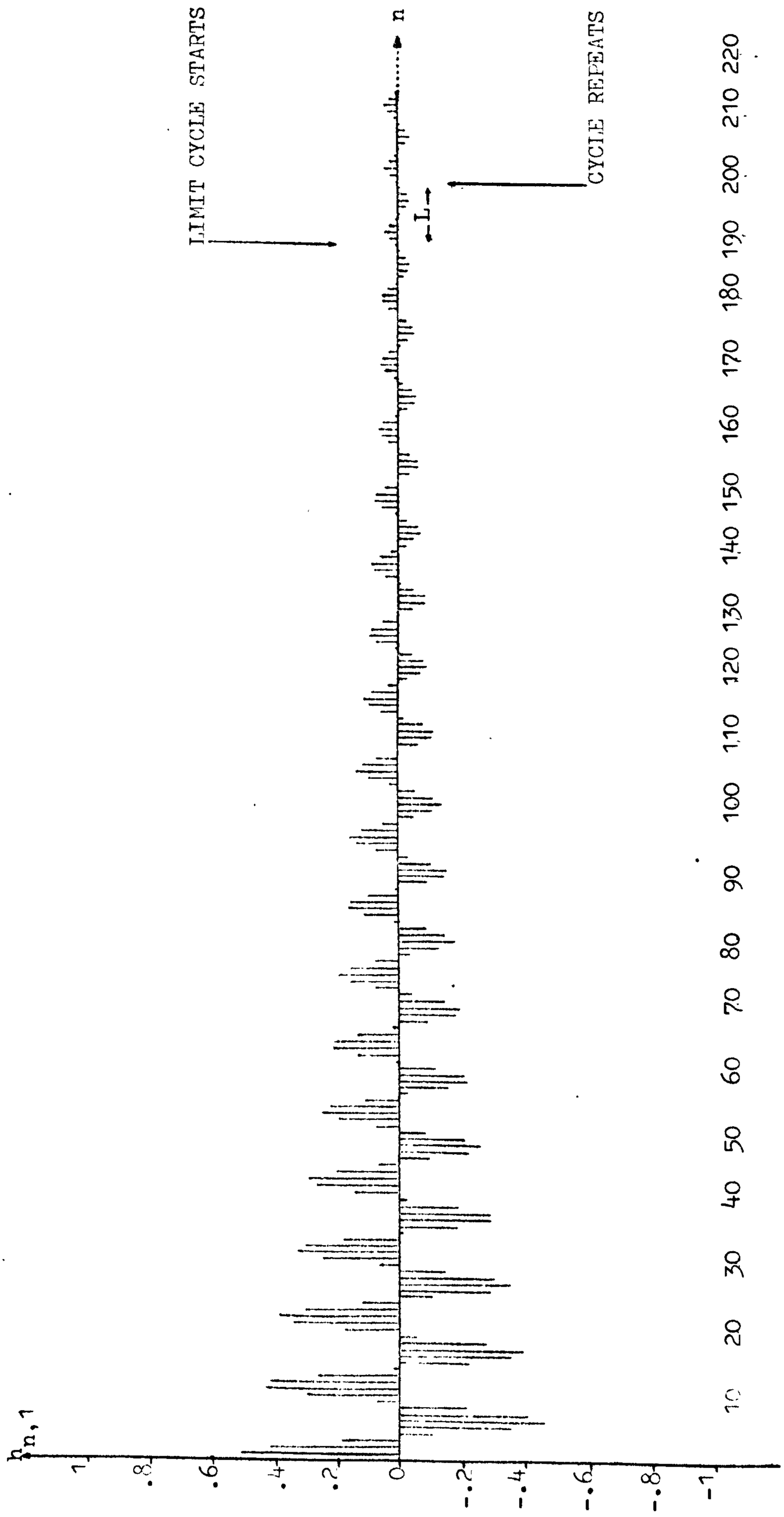


FIG: 8.2.1 EXPERIMENTAL IMPULSE RESPONSE OF THE LOWER TRANSITION SECOND-ORDER ELLIPTICAL, FILTER $H_1^T(z)$ OF BANDPASS FILTER A (WORDLENGTH: 8 BITS INCLUDING SIGN).

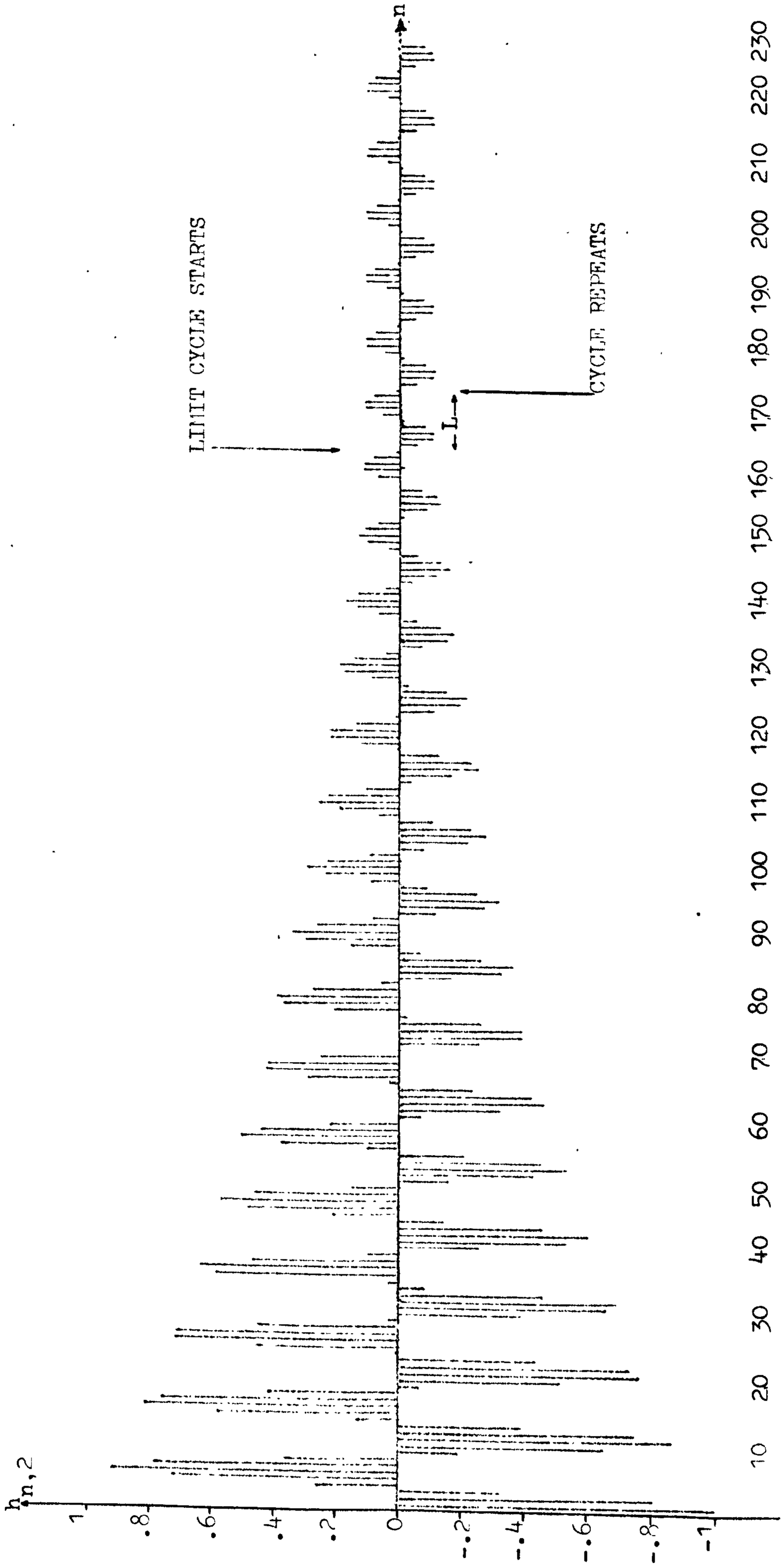


FIG: 8.2.2 EXTREMENTAL IMPULSE RESPONSE OF THE FIRST SECOND-ORDER PASSBAND ELEMENTAL
OF BANDPASS FILTER A (WORDLENGTH: 8 BITS INCLUDING SIGN).

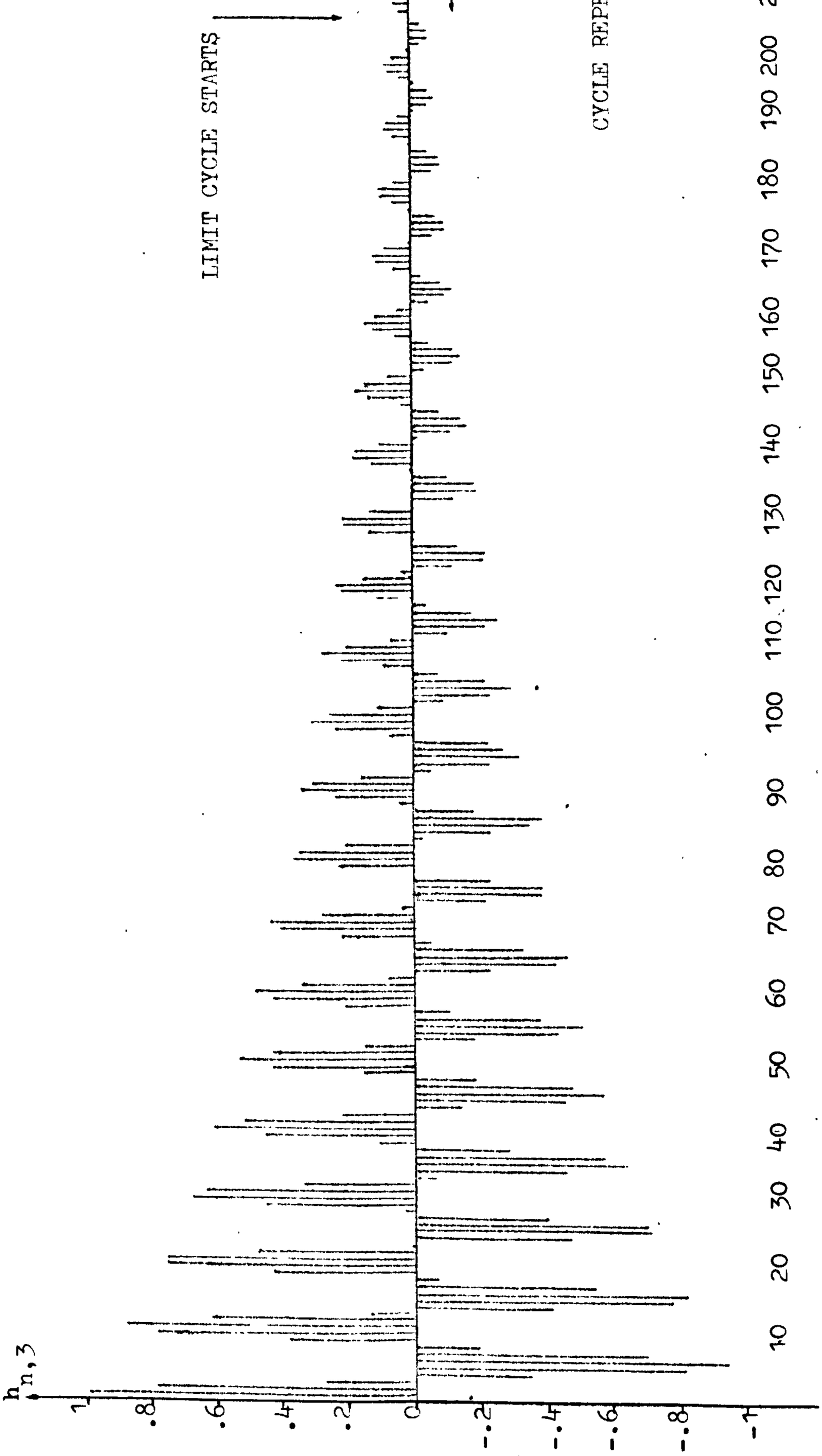


FIG: 8.2.3 EXPERIMENTAL IMPULSE RESPONSE OF THE SECOND SECOND-ORDER PASSBAND FILTER $H_2^P(z)$
OF BANDPASS FILTER A (WORDLENGTH: 8 BITS INCLUDING SIGN).

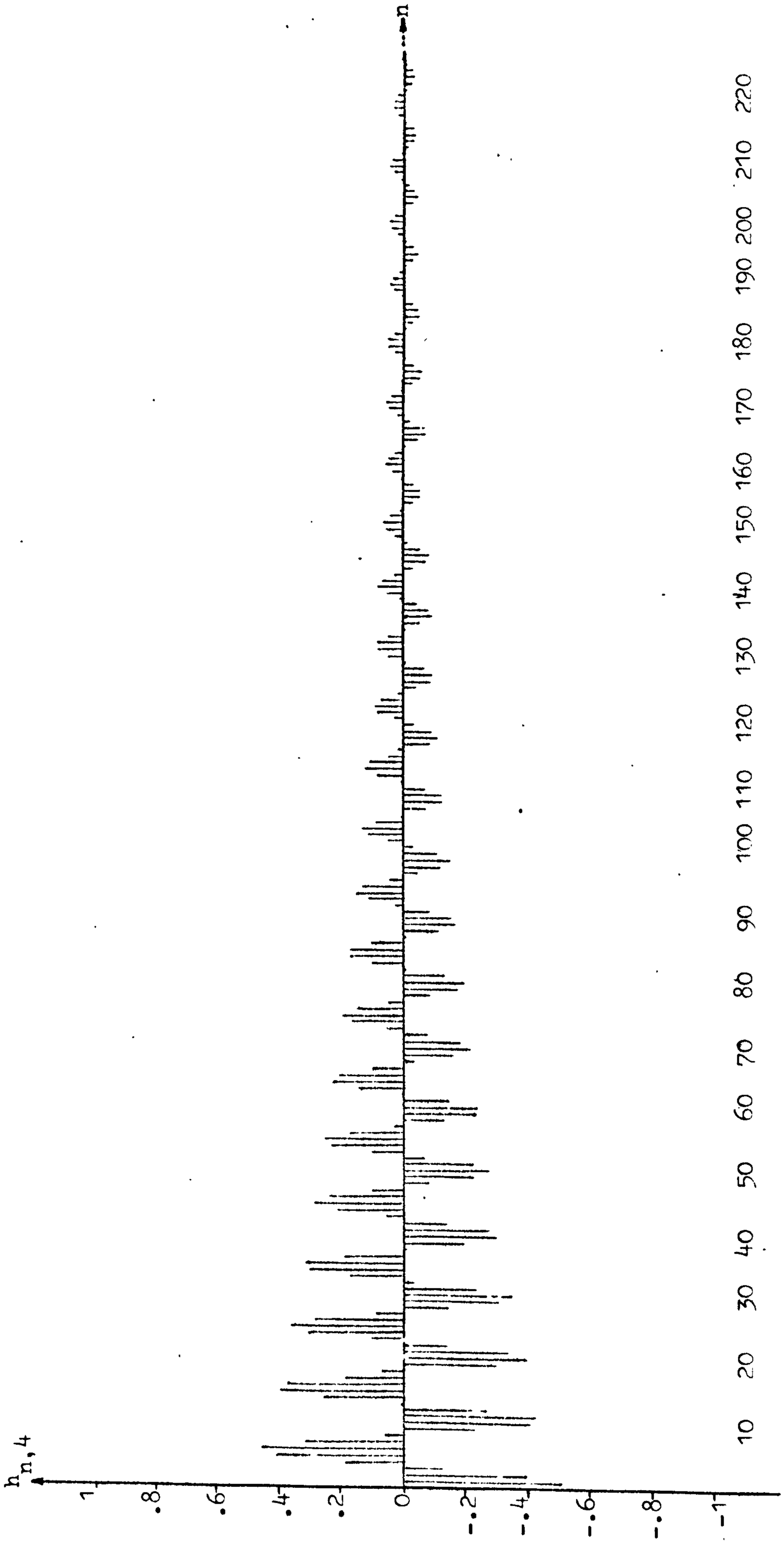
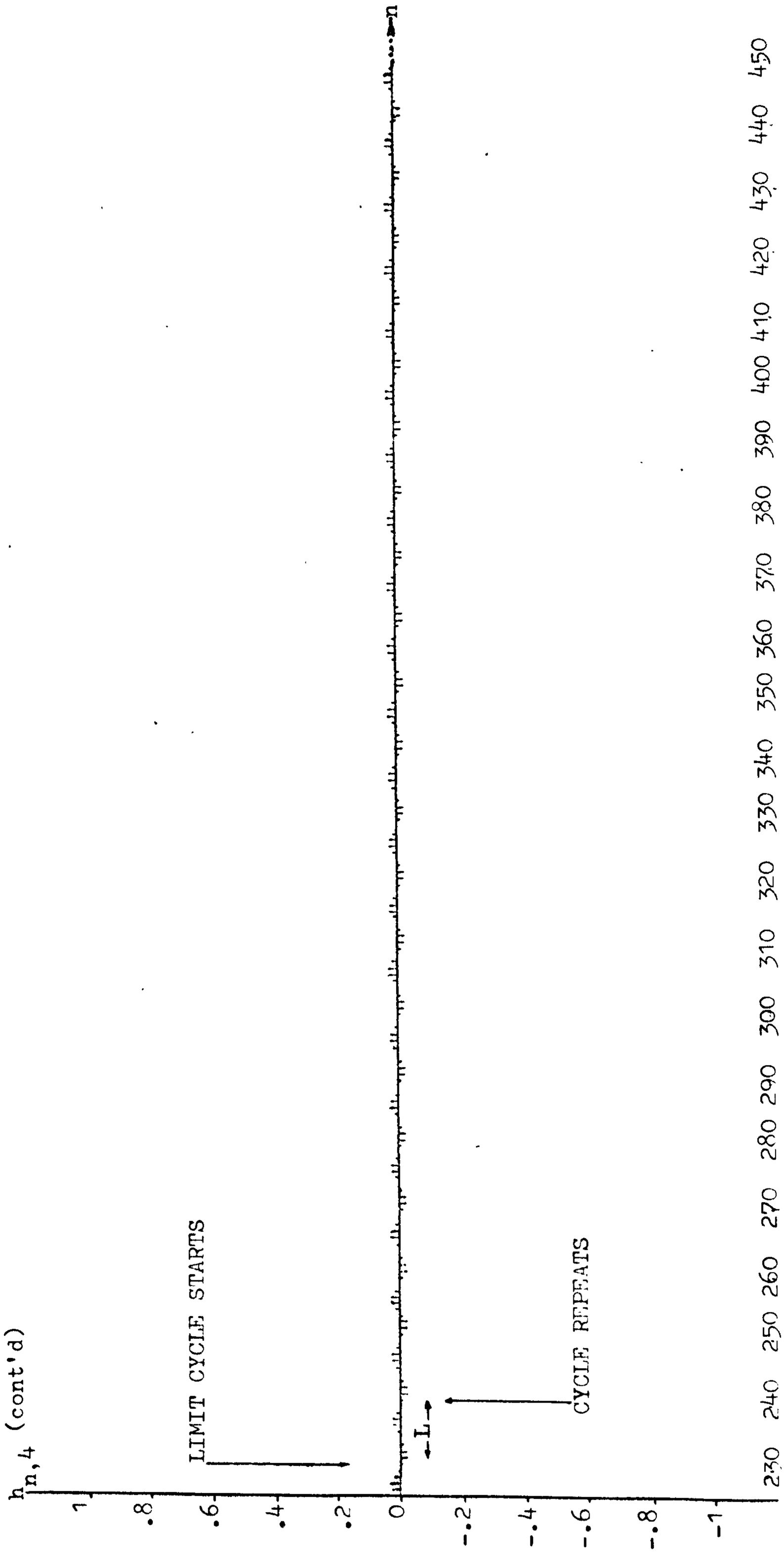


FIG: 8.2.4 EXPERIMENTAL IMPULSE RESPONSE OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER $-H_u^T(z)$
OF BANDPASS FILTER A (WORDLENGTH: 8 BITS INCLUDING SIGN).



CONTINUATION OF FIGURE 8.2.4.

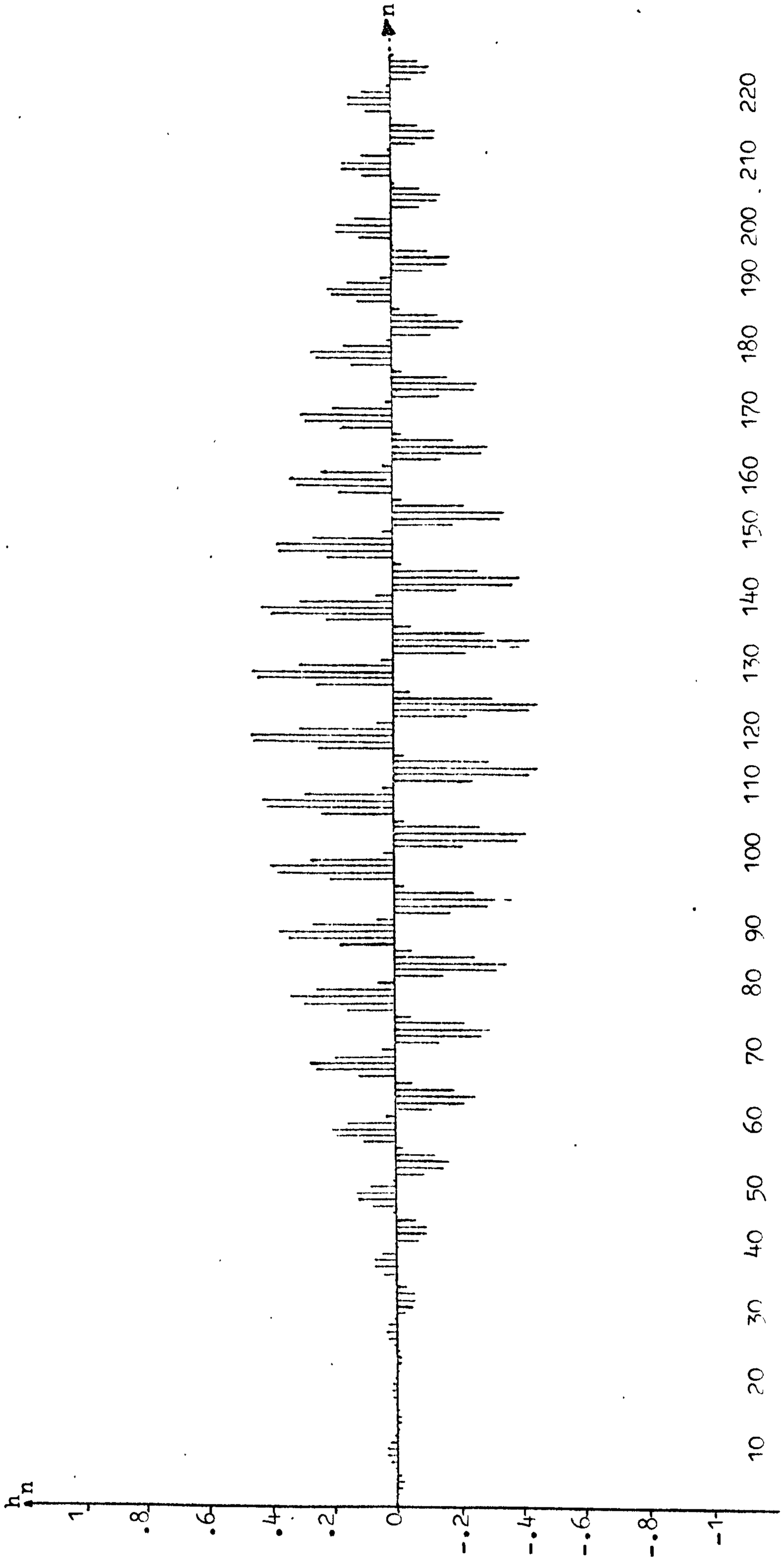
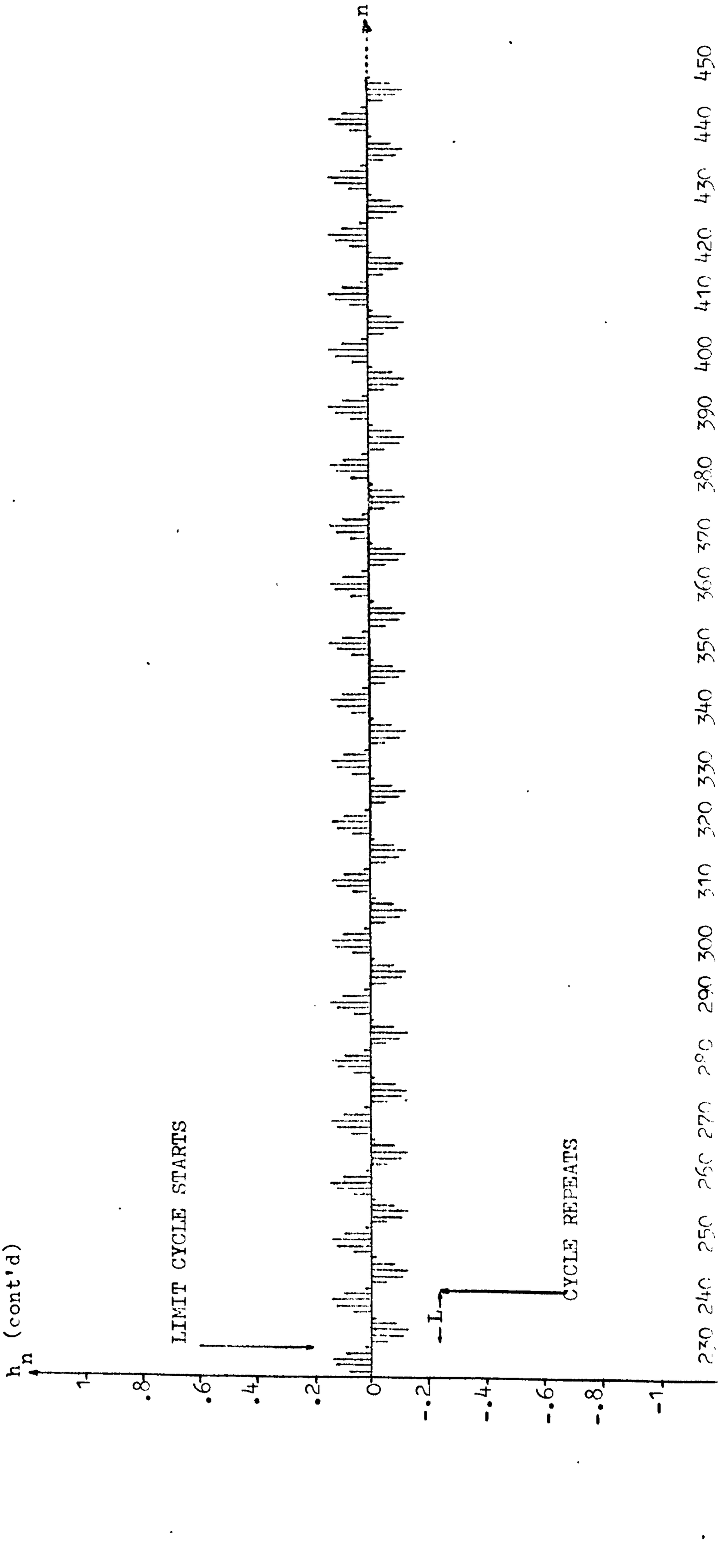


FIG: 8.2.5 EXPERIMENTAL IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER A REALIZED IN CONFIGURATION (b) IN FIGURE 8.1.2 WITH THE OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.5 (WORDLENGTH: 8 BITS INCLUDING SIGN).



CONTINUATION OF FIGURE 8.2.5

Next, we consider equation (8.2.2) realized in configuration (a) shown in figure 8.1.2 previously. Now, equation (8.2.2) can be rewritten as follows:

$$H_Q(z) = \frac{(0.5 - 0.4067077903z^{-1})}{1 - 1.626831161z^{-1} + 0.9741515374z^{-2}}$$

$$- \frac{(1 - 0.80603030 48z^{-1})}{1 - 1.61206061z^{-1} + 0.9741515374z^{-2}}$$

$$+ \frac{(1 - 0.790832529z^{-1})}{1 - 1.5816665058z^{-1} + 0.9741515374z^{-2}}$$

$$- \frac{(0.5 - 0.3915252775z^{-1})}{1 - 1.56610111z^{-1} + 0.9741515374z^{-2}} \qquad (8.2.3)$$

with the quantized value of W_n as 0.5. The respective experimental impulse responses of the four second-order elemental filters and that of the resultant eighth-order bandpass filter are shown in figures 8.2.6 to 8.2.10. Again, for the purpose of reference, we shall refer to this eighth-order bandpass filter realized in configuration (a) in figure 8.1.2, with optimized transition sample W_n quantized to a value of 0.5, as bandpass filter B. The value of the unit input impulse was 0.1111111=0.9921875, while the respective contents of the CBFCS memories for the four second-order elemental filters are shown in tables C5 to C8 in appendix C.

Figure 8.2.6 shows the experimental impulse response $h_{n,1}$ of the second-order lower transition elemental filter $H_1^T(z)$ of the above bandpass filter B. The scaling by the quantized value of W_n was precomputed and performed within the CBFCS memory C5. It is noticed from figure 8.2.6 that the impulse response $h_{n,1}$ dies down to a limit cycle behaviour at the 160th sample, which is somewhat sooner when compared with the impulse response $h_{n,1}$ of bandpass filter A shown previously in figure 8.2.1 which dies down to a limit cycle behaviour at the 187th sample. The limit cycle has a period L=11 so that it repeats every eleven samples. Furthermore, the maximum amplitude of this limit cycle is seven quantization steps compared with five in

the case of the limit cycle of $h_{n,1}$ of bandpass filter A. Thus, the ratio of the maximum amplitude of this limit cycle to the dynamic range is 7/128=0.0546875 or approximately $5\frac{1}{2}\%$. Hence, all the above mentioned has made the impulse response $h_{n,1}$ of bandpass filter B less desirable than that of bandpass filter A. Finally, it is also observed that the limit cycle behaviour of $h_{n,1}$ is approximately sinusoidal, and again, had there been no finite word length effects, this impulse response would have died down to zero value eventually without any limit cycle behaviour, as discussed in appendix B.

Figure 8.2.7 shows the experimental impulse response $h_{n,2}$ of the first second-order passband elemental filter $-H_1^P(z)$ of the above bandpass filter B. As in the previous case, the minus sign of the scaling factor -1 of the above elemental filter is absorbed within the CBFCS memory C6 in the precomputed compensated partial products. It is seen from figure 8.2.7 that the impulse response $h_{n,2}$ dies down to a limit cycle at the 162nd sample. In fact, $h_{n,2}$ in the case of bandpass filter B is the same as that in the case of bandpass filter A. The limit cycle has a period L=10 so that it repeats every ten samples. Furthermore, this limit cycle has a maximum amplitude of fourteen quantization steps, such that the ratio of its maximum amplitude to the dynamic range is therefore 14/128 or nearly 11%. It will be seen later that this is the largest of all four limit cycles of the four elemental filters the constituted the above bandpass filter B. In fact, its presence has contributed largely to the resultant amplitude of the limit cycle of the above eighth-order bandpass filter B. It is also observed that this limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour, as discussed in appendix B.

Figure 8.2.8 shows the experimental impulse response $h_{n,3}$ of the second second-order passband elemental filter $H_2^P(z)$ of the above bandpass filter B. It is seen from figure 8.2.8 that the impulse response $h_{n,3}$ dies down to a limit cycle at the 204th sample. This limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is six quantization steps, such that the ratio of its maximum amplitude to the dynamic range is

therefore 6/128=0.046875 or just about 4½%. It is also noticed that the nature of this limit cycle is approximately sinusoidal. In fact, $h_{n,3}$ in the case of bandpass filter B is the same as that in the case of bandpass filter A. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour, as discussed in appendix B.

Figure 8.2.9 shows the experimental impulse response $h_{n,4}$ of the second-order upper transition elemental filter $-H_u^T(z)$ of the above bandpass filter B. The scaling by the quantized value of W_n was precomputed and performed within the CBFCS memory C8, with the minus sign of the above elemental filter taken into account. It is observed that the impulse response $h_{n,4}$ dies down to a limit cycle at the 190th sample. The limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is six quantization steps, such that the ratio of its maximum amplitude to the dynamic range is therefore 6/128=0.046875 or just about 4½%. Thus, the impulse response $h_{n,4}$ of bandpass filter B is less desirable than that of bandpass filter A which has only a maximum amplitude of three quantization steps and dies down to a limit cycle at the 230th sample. Finally, it is also observed the above limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour as discussed in appendix B.

Figure 8.2.10 shows the experimental impulse response h_n of the eighth-order bandpass filter B realized in configuration (a) shown in figure 8.1.2 with the optimized transition sample W_n quantized to a value of 0.5 and a wordlength of eight bits including sign. As in the case of bandpass filter A, it is seen from figures 8.2.6 to 8.2.10 that h_n of bandpass filter B is the summation of $h_{n,1}$, $h_{n,2}$, $h_{n,3}$ and $h_{n,4}$. From figure 8.2.10, the impulse response h_n is seen to die down to a limit cycle behaviour at the 206th sample, with a period L=110, such that it repeats every 110 samples. As mentioned before, the period of the limit cycle of bandpass filter B is the least common multiple (LCM) of the periods of the limit cycles of the individual second-order elemental filters (note that L=11 for the first limit

cycle and L=10 for the rest three limit cycles). Although a certain amount of limit cycle cancellation is observed as mentioned before, the maximum amplitude of the limit cycle of the above bandpass filter B is twenty-four quantization steps, such that the ratio of its maximum amplitude to the dynamic range is 24/128=0.1875 or nearly 19%. The magnitude of such a limit cycle is largely due to the magnitude of the limit cycle of $-H_1^P(z)$ which amounts to fourteen quantization steps alone. In addition, owing to the fact that the periods of the four limit cycles of the four second-order elemental filters are unequal, giving a large least common multiple (LCM) of a value L=110, these limit cycles tend to beat into a complex waveform, as shown in the latter part of figure 8.2.10, with a relatively larger magnitude. Again, had there been no finite word length effects, the impulse response h_n would have died down to zero value eventually without any limit cycle behaviour (see a later section).

At this juncture, some comments are now in order. So far, we have discussed two realizations (shown in figure 8.1.2) of the design of a 4-pole, eighth-order bandpass filter centred at 120Hz. The above two realization configurations have also been referred to as bandpass filter A and bandpass filter B for reference purposes. In bandpass filter B, which was realized in configuration (a) shown in figure 8.1.2, the scaling by the optimized transition sample W_n was performed within the respective CBFCS memories, and the compensated partial products within the respective CBFCS memories were precomputed according to equation (8.2.3). In this manner, the unit input impulse was first scaled by the value of W_n and there was, in addition, a loss in signal-to-noise ratio due to input scaling. Since the contents of the CBFCS memories of bandpass filter B is different from those of bandpass filter A, the initial conditions for zero-input limit cycles have changed which accounted for the different limit cycle behaviours of $h_{n,1}$ and $h_{n,4}$ of bandpass filters A and B respectively. It has also been concluded that the impulse responses of bandpass filter A and its constituent elemental filters are more desirable than the corresponding set of impulse responses of bandpass filter B and its constituent elemental filters, especially when limit cycle behaviour is concerned. This may be explained by the reason that in bandpass

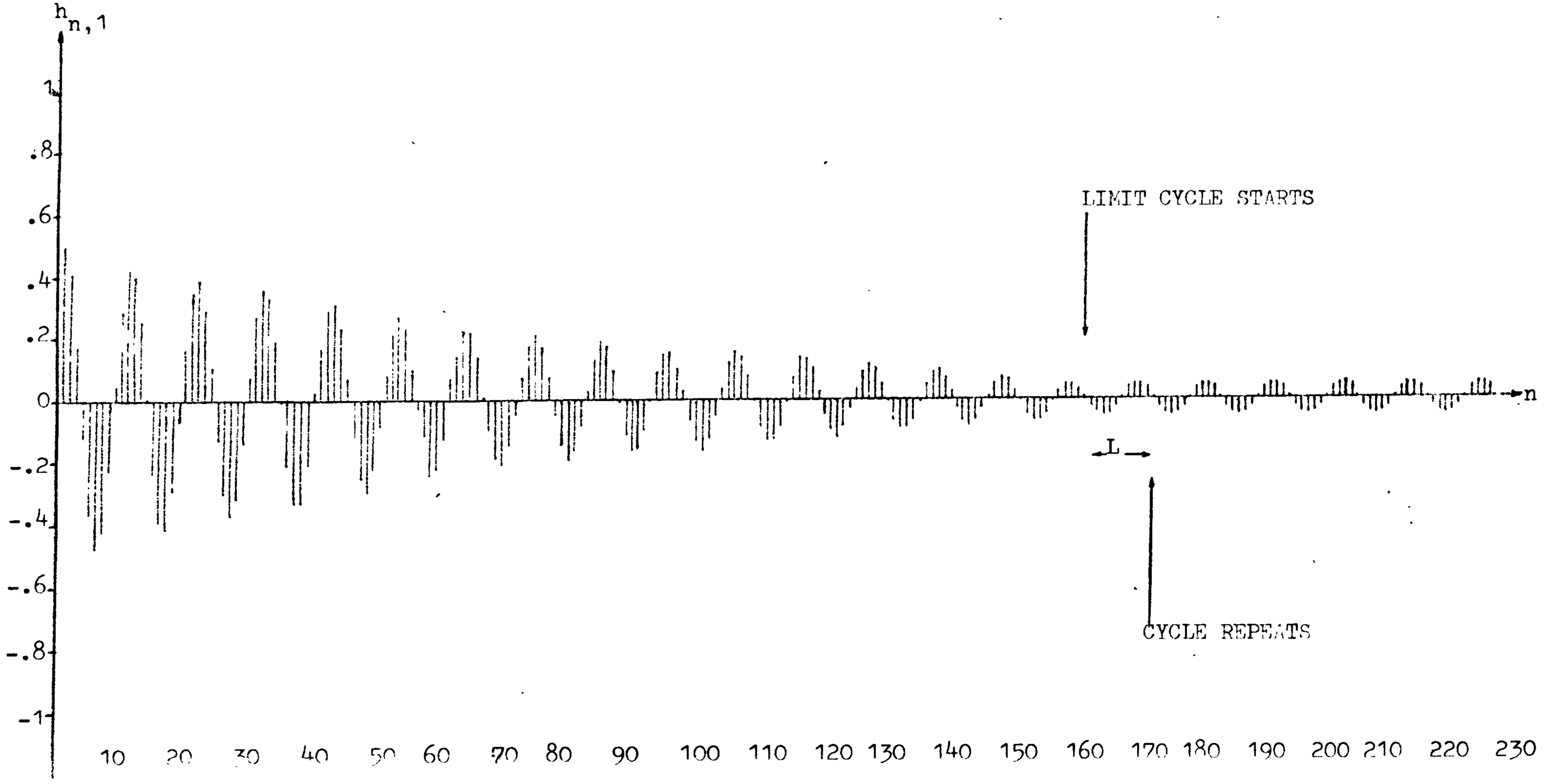


FIG: 8.2.6 EXPERIMENTAL IMPULSE RESPONSE OF THE LOWER TRANSITION SECOND-ORDER ELEMENTAL FILTER $H_1^T(z)$ OF BANDPASS FILTER B (WORDLENGTH: 8 BITS INCLUDING SIGN).

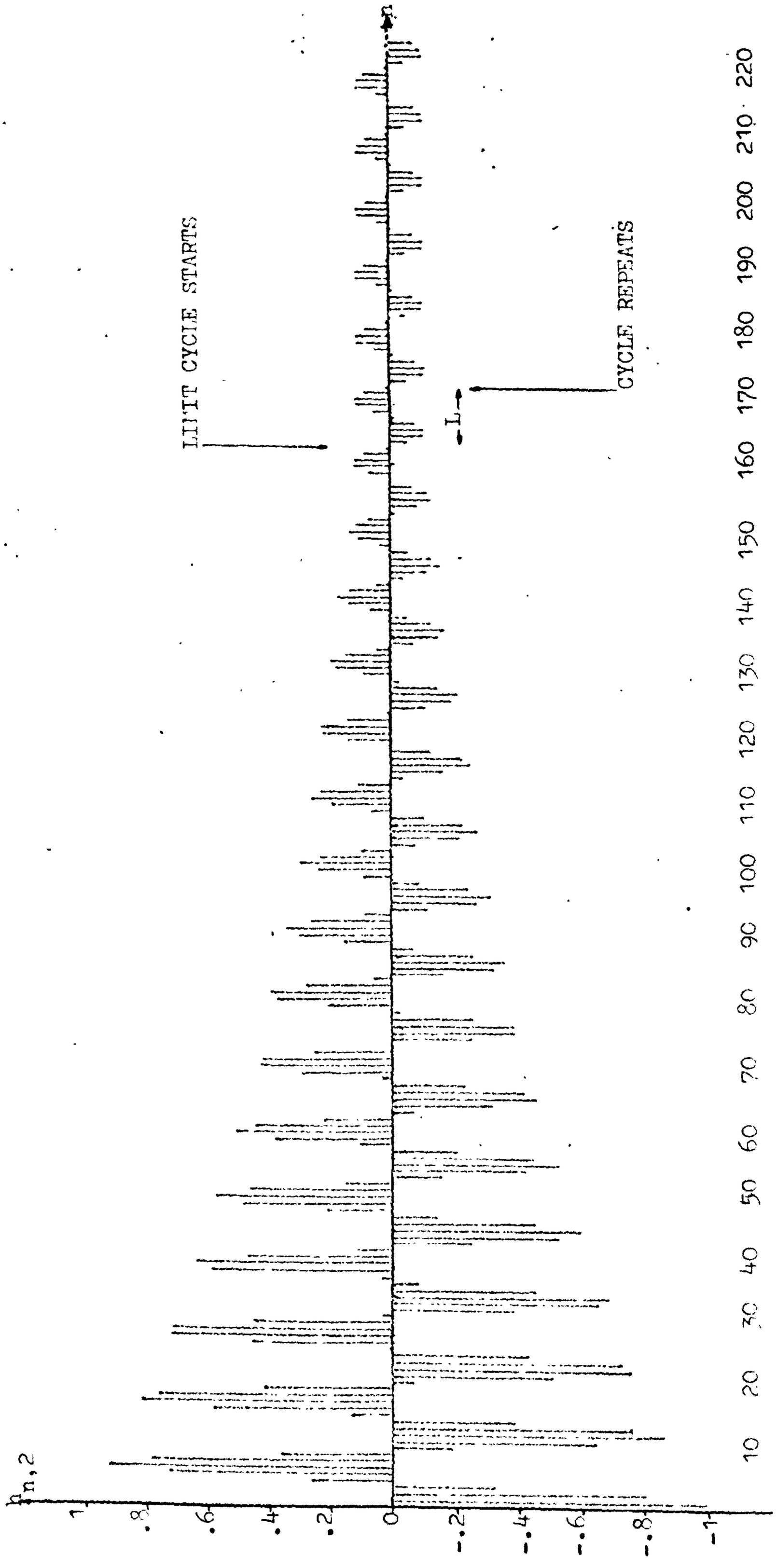


FIG: 8.2.7 EXPERIMENTAL IMPULSE RESPONSE OF THE FIRST SECOND-ORDER PASSBAND ELEMENTAL FILTER $-H_1^P(z)$ OF BANDPASS FILTER B (WORDLENGTH: 8 BITS INCLUDING SIGN).

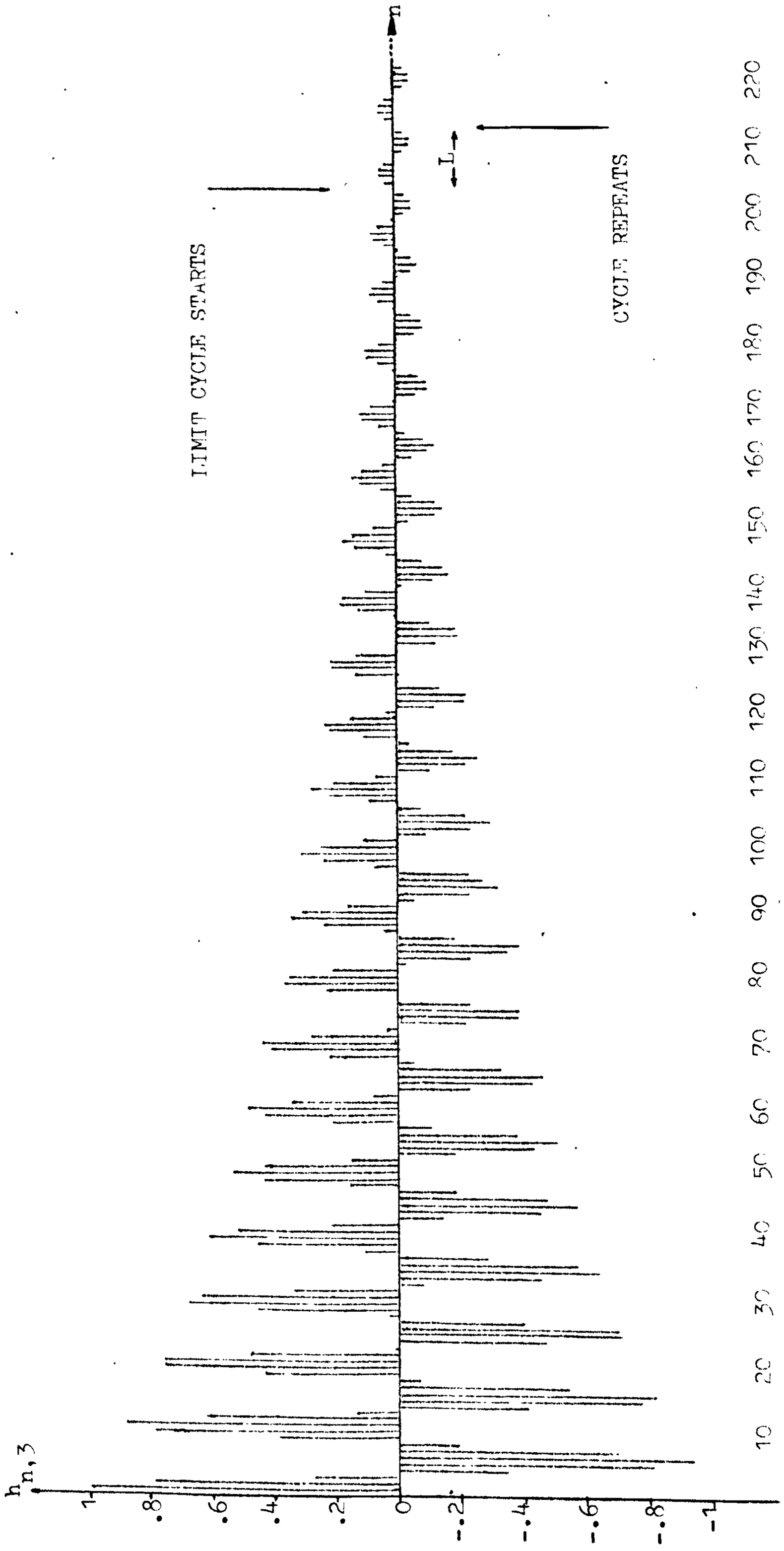


FIG: 8.2.8 EXPERIMENTAL IMPULSE RESPONSE OF THE SECOND-ORDER PASSBAND ELEMENTAL FILTER $H_2^P(z)$
OF BANDPASS FILTER B (WORDLENGTH: 8 BITS INCLUDING SIGN).

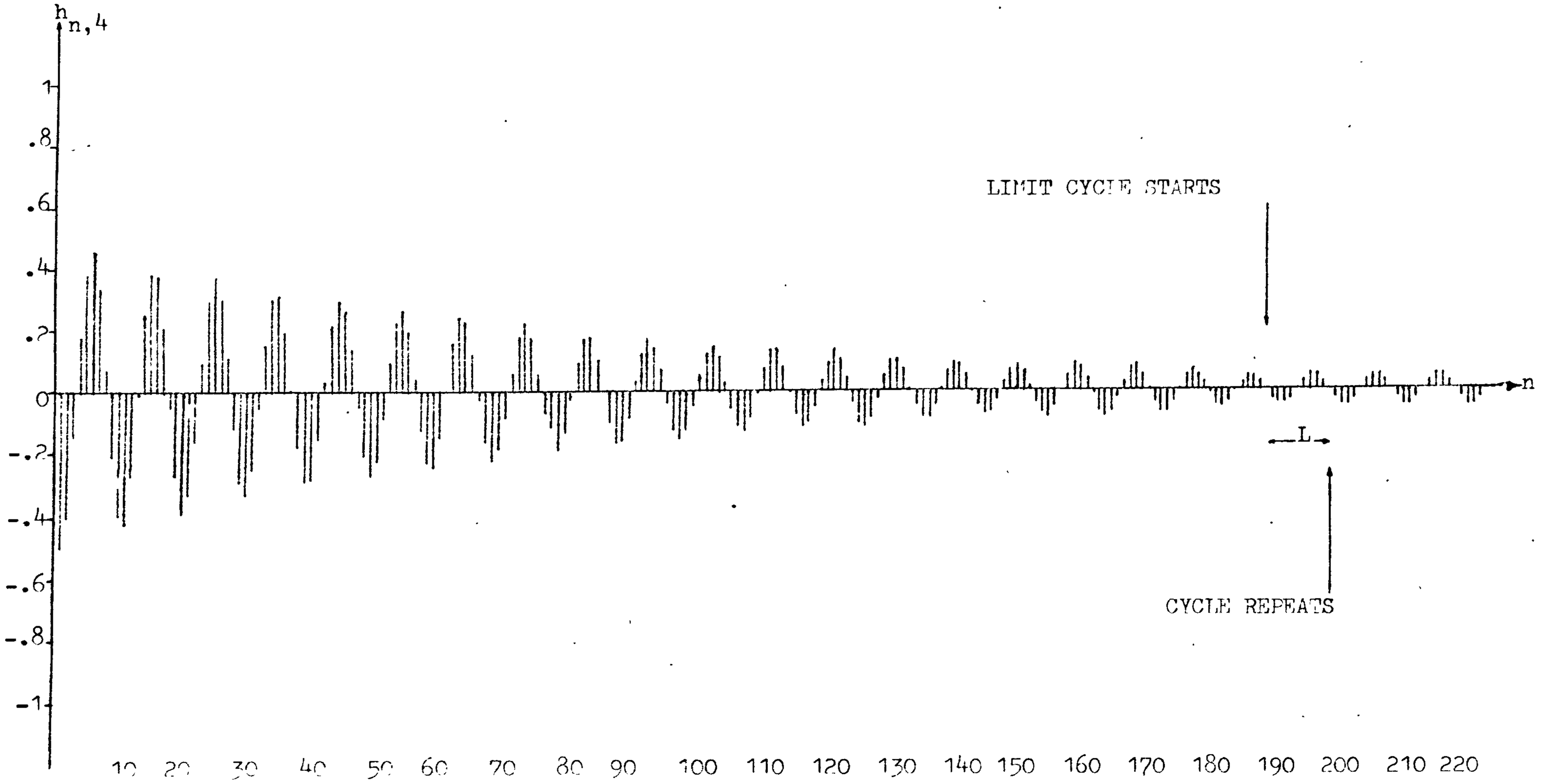


FIG: 8.2.9 EXPERIMENTAL IMPULSE RESPONSE OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER $-H_u^T(z)$ OF BANDPASS FILTER B (WORDLENGTH: 8 BITS INCLUDING SIGN).

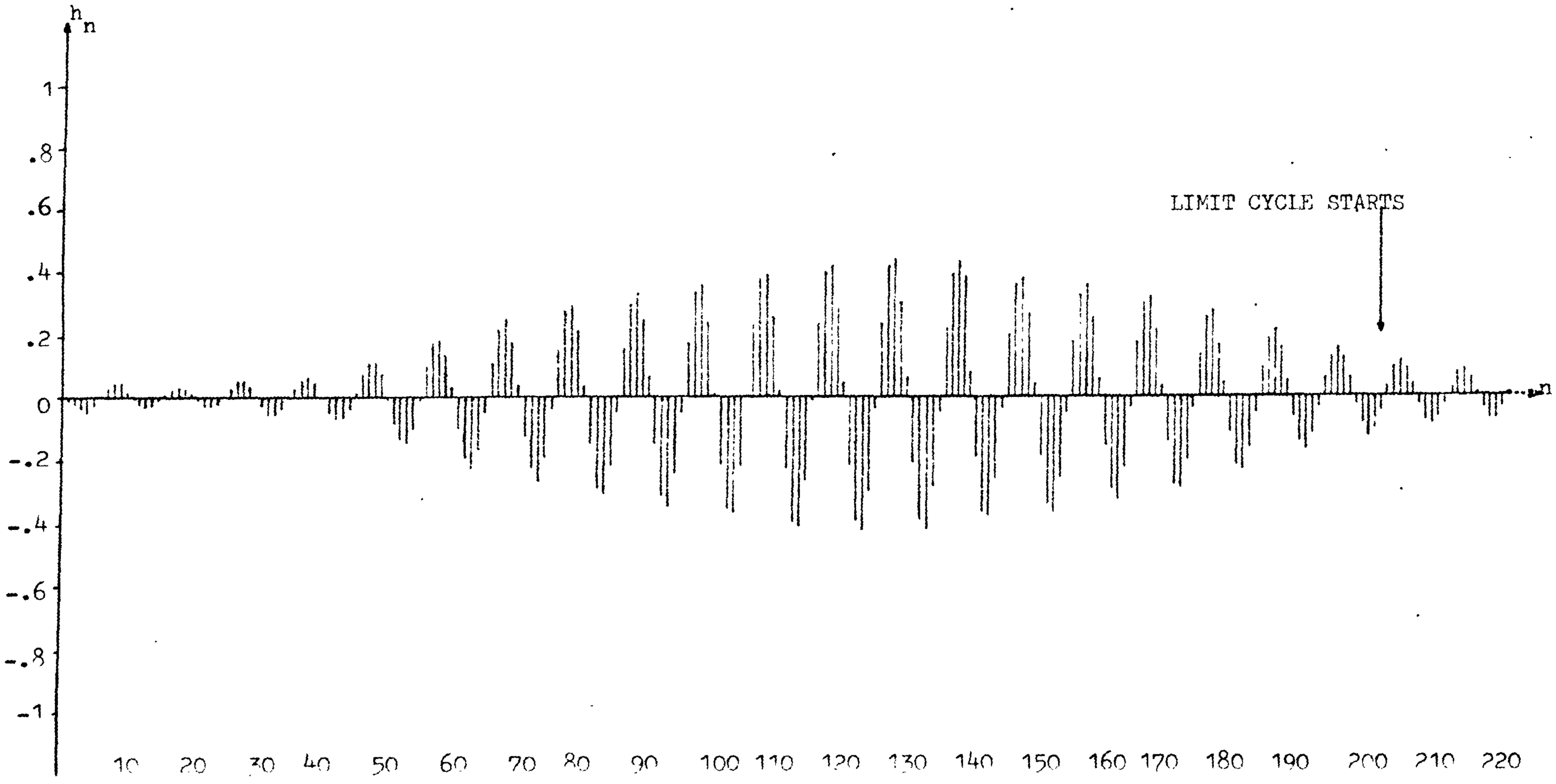


FIG: 8.2.10 EXPERIMENTAL IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER B REALIZED IN CONFIGURATION (a) IN FIGURE 8.1.2 WITH THE OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.5 (WORDLENGTH: 8 BITS INCLUDING SIGN).

CONTINUATION OF FIGURE 8.2.10

filter A, which was realized in configuration (b) shown in figure 8.1.2, the limit cycles of the two transition elemental filters are scaled by the value of W_n since the scaling was performed at the output summation. Finally, it has been observed that the impulse response h_n of bandpass filter B is of a shorter duration, with regards to the occurrence of limit cycle behaviour, than that of bandpass filter A.

Further to our experimental investigation into the choice of realization configurations (a) and (b) shown in figure 8.1.2, we shall examine configuration (a) again with a slightly different design. Consider bandpass filter B with a change in the value of W_n. This time, we consider a non-optimum value of 0.4149780277 so that the bandpass filter is no longer optimized for the steepest skirt (when the value of W_n was 0.5). Again, for the purpose of reference, we shall refer to this filter as bandpass filter C. This filter has its quantized transfer function written as follows:

$$H_Q(z) = \frac{(0.4149780277 - 0.3375495933z^{-1})}{1 - 1.626831161z^{-1} + 0.97415153 74z^{-2}}$$

$$- \frac{(1 - 0.8060303048z^{-1})}{1 - 1.61206061z^{-1} + 0.9741515374z^{-2}}$$

$$+ \frac{(1 - 0.790832529z^{-1})}{1 - 1.581665058z^{-1} + 0.9741515374z^{-2}}$$

$$- \frac{(0.4149780277 - 0.3249487749z^{-1})}{1 - 1.56610111z^{-1} + 0.9741515374z^{-2}} \qquad (8.2.4)$$

with the quantized non-optimized value of W_n as 0.4149780277 to sixteen bits accuracy including sign. The respective experimental impulse responses of the four second-order elemental filters and that of the resultant eighth-order bandpass filter C are shown in figures 8.2.11 to 8.2.15. The value of the unit input impulse was 0.1111111 = 0.9921875, while the respective contents of the CBFCS memories for the four second-order elemental filters are shown in tables C9 to C12 in appendix C.

Figure 8.2.11 shows the experimental impulse response $h_{n,1}$ of the second-order lower transition elemental filter $H_1^T(z)$ of the above bandpass filter C. The scaling by the quantized value of the non-optimized W_n was precomputed and performed within the CBFCS memory C9. It is noticed from figure 8.2.11 that the impulse response $h_{n,1}$ dies down to a limit cycle behaviour at the 145th sample. The limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is ten quantization steps such that the ratio of its maximum amplitude to the dynamic range 10/128 =0.078125 or just under 8%. Like bandpass filter B, the magnitude of the above limit cycle is not being scaled by the value of W_n since both bandpass filters B and C were realized in configuration (a) shown in figure 8.1.2. Finally, it is also observed that the limit cycle behaviour of $h_{n,1}$ is approximately sinusoidal , and again, had there been no finite word length effects, this impulse response would have died down to zero value eventually without any limit cycle behaviour, as discussed in appendix B.

Figure 8.2.12 shows the experimental impulse response $h_{n,2}$ of the first second-order passband elemental filter $-H_1^P(z)$ of the above bandpass filter C. It is seen from figure 8.2.12 that the impulse response $h_{n,2}$ dies down to a limit cycle at the 162nd sample. In fact, $h_{n,2}$ in the case of bandpass filter C is the same as that in the case of bandpass filter B as expected. The limit cycle has a period L=10 and a maximum amplitude of fourteen quantization steps, so that the ratio of its maximum amplitude to the dynamic range is 14/128 = 0.109375 or nearly 11%. It is also observed that this limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value without any limit cycle behaviour as discussed in appendix B.

Figure 8.3.13 shows the experimental impulse response $h_{n,3}$ of the second second-order passband elemental filter $H_2^P(z)$ of the above bandpass filter C. It is seen from figure 8.2.13 that the impulse response $h_{n,3}$ dies down to a limit cycle at the 204th sample. This limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is six quantization steps such that the ratio of its maximum amplitude to the dynamic range is

therefore 6/128=0.046875 or just about 4½%. It is also noticed that the nature of this limit cycle is approximately sinusoidal. In fact, $h_{n,3}$ in the case of bandpass filter C is the same as that in the case of bandpass filter B as expected. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour as discussed in appendix B.

Figure 8.2.14 shows the experimental impulse response $h_{n,4}$ of the second-order upper transition elemental filter $-H_u^T(z)$ of the above bandpass filter C. The scaling by the quantized value of the non-optimized transition sample W_n was precomputed and performed within the CBFCS memory C12, with the minus sign of the above elemental filter taken into account. It is observed that the impulse response $h_{n,4}$ dies down to a limit cycle at the 142nd sample. The limit cycle has a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is six quantization steps, such that the ratio of its maximum amplitude to the dynamic range is therefore 6/128=0.046875 or just about 4½%. Finally, it is also observed that the above limit cycle is approximately sinusoidal in nature. Again, had there been no finite word length effects, the above impulse response would have died down to zero value eventually without any limit cycle behaviour as discussed in appendix B.

Figure 8.2.15 shows the experimental impulse response h_n of the eighth-order bandpass filter C realized in configuration (a) shown in figure 8.1.2, with the non-optimized transition sample W_n quantized to a value of 0.4149780277 and a wordlength of eight bits including sign. As in the cases of bandpass filters A and B, it is seen from figures 8.2.11 to 8.2.15 that h_n of bandpass filter C is the summation of $h_{n,1}$, $h_{n,2}$, $h_{n,3}$ and $h_{n,4}$. From figure 8.2.15, the impulse response h_n is seen to die down to a limit cycles behaviour at the 206th sample, with a period L=10, such that it repeats every ten samples. As mentioned before, the period of the limit cycle of the above bandpass C is the least common multiple (LCM) of the periods of the limit cycles of the individual second-order elemental filters. Although a certain amount of limit cycle cancellation is observed, the maximum amplitude of the limit cycle of the above eighth-order bandpass filter C is twenty-three quantization steps, such that the

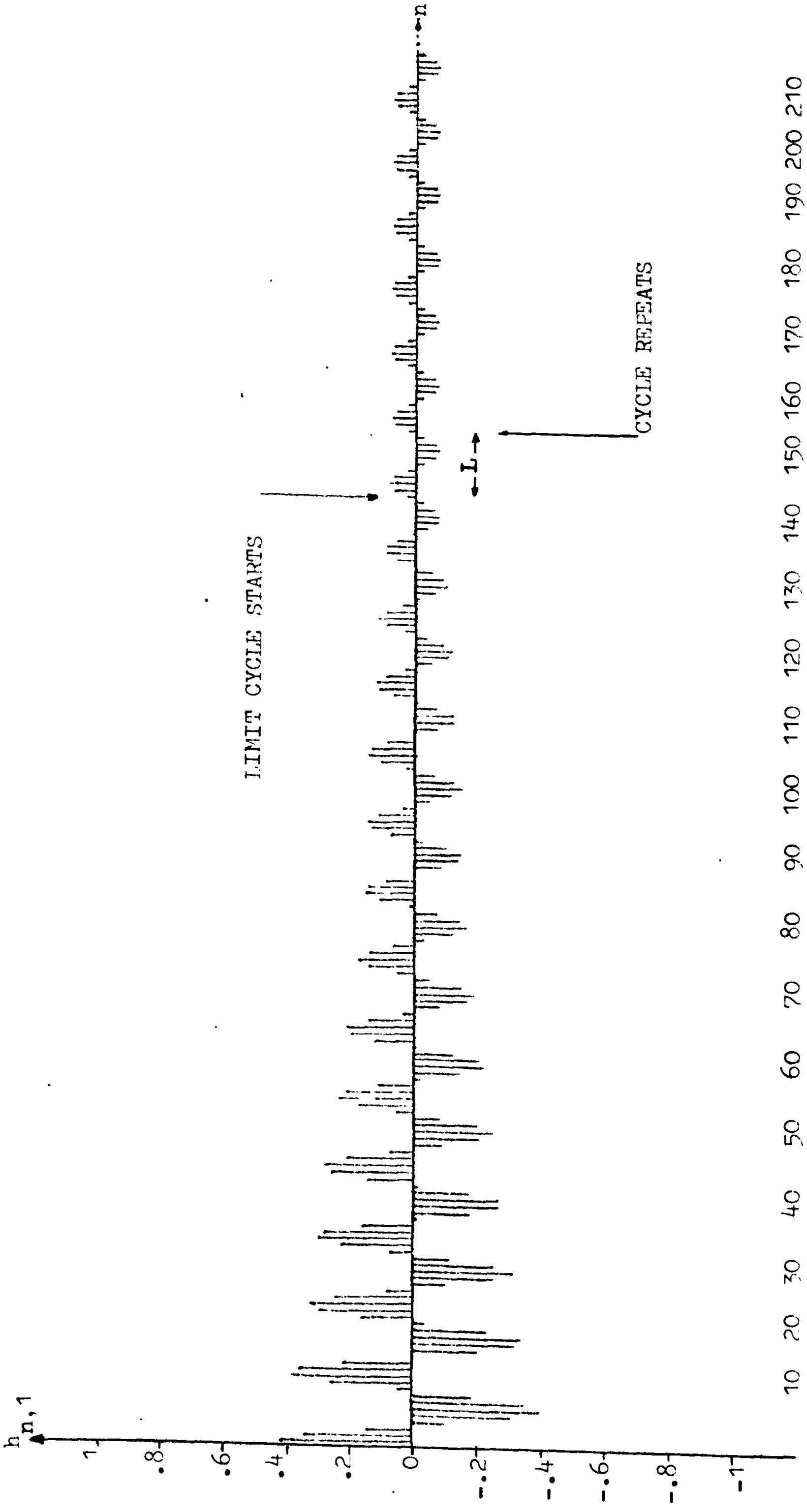


FIG: 8.2.11 EXPERIMENTAL IMPULSE RESPONSE OF THE LOWER TRANSITION SECOND-ORDER ELEMENTAL FILTER $H_1^T(z)$ OF BANDPASS FILTER C (WORDLENGTH: 8 BITS INCLUDING SIGN).

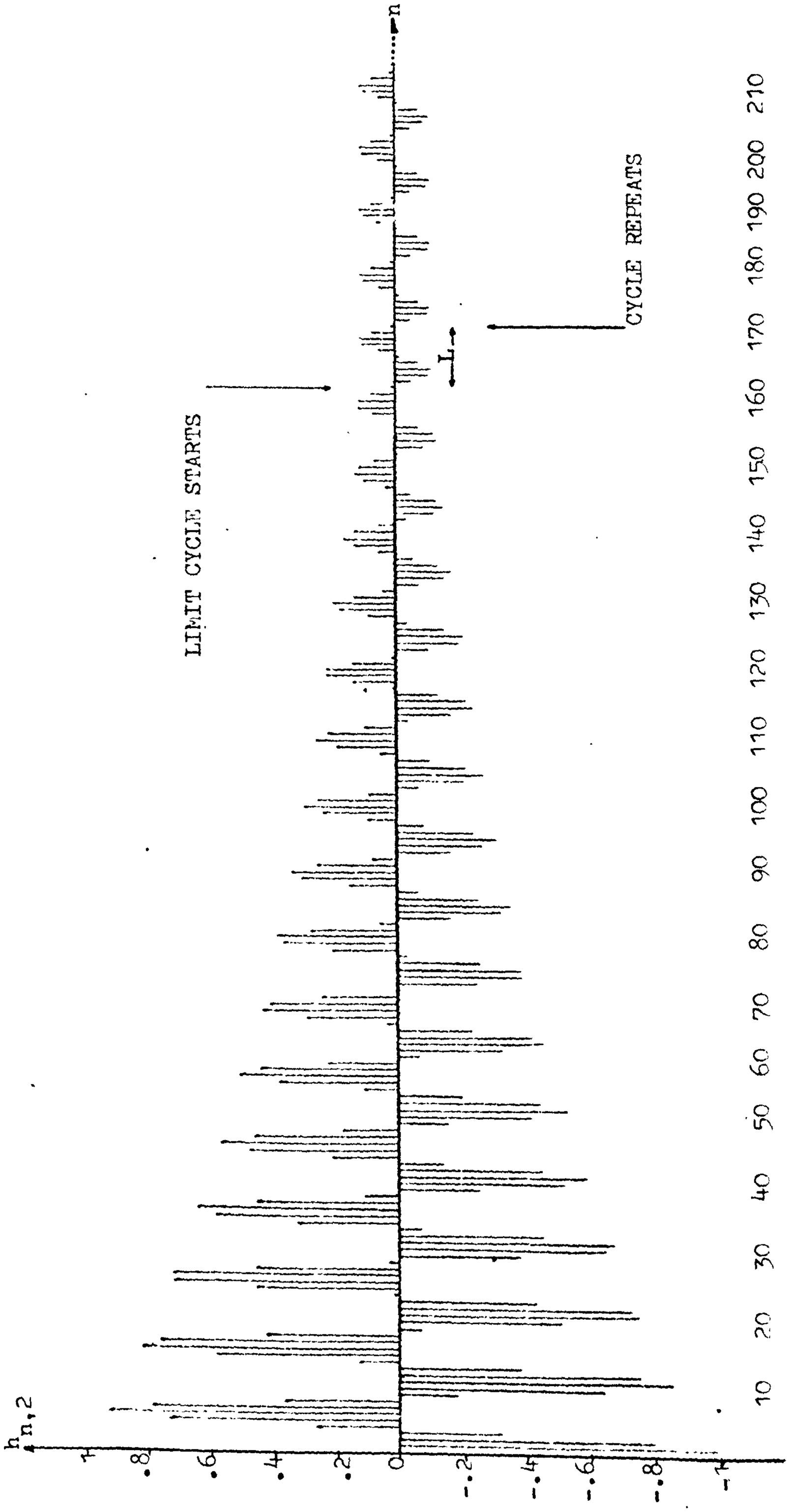


FIG: 8.2.12 EXPERIMENTAL IMPULSE RESPONSE OF THE FIRST SECOND-ORDER PASSBAND ELEMENTAL FILTER $-H_1^P(z)$ OF BANDPASS FILTER C (WORDLENGTH: 8 BITS INCLUDING SIGN).

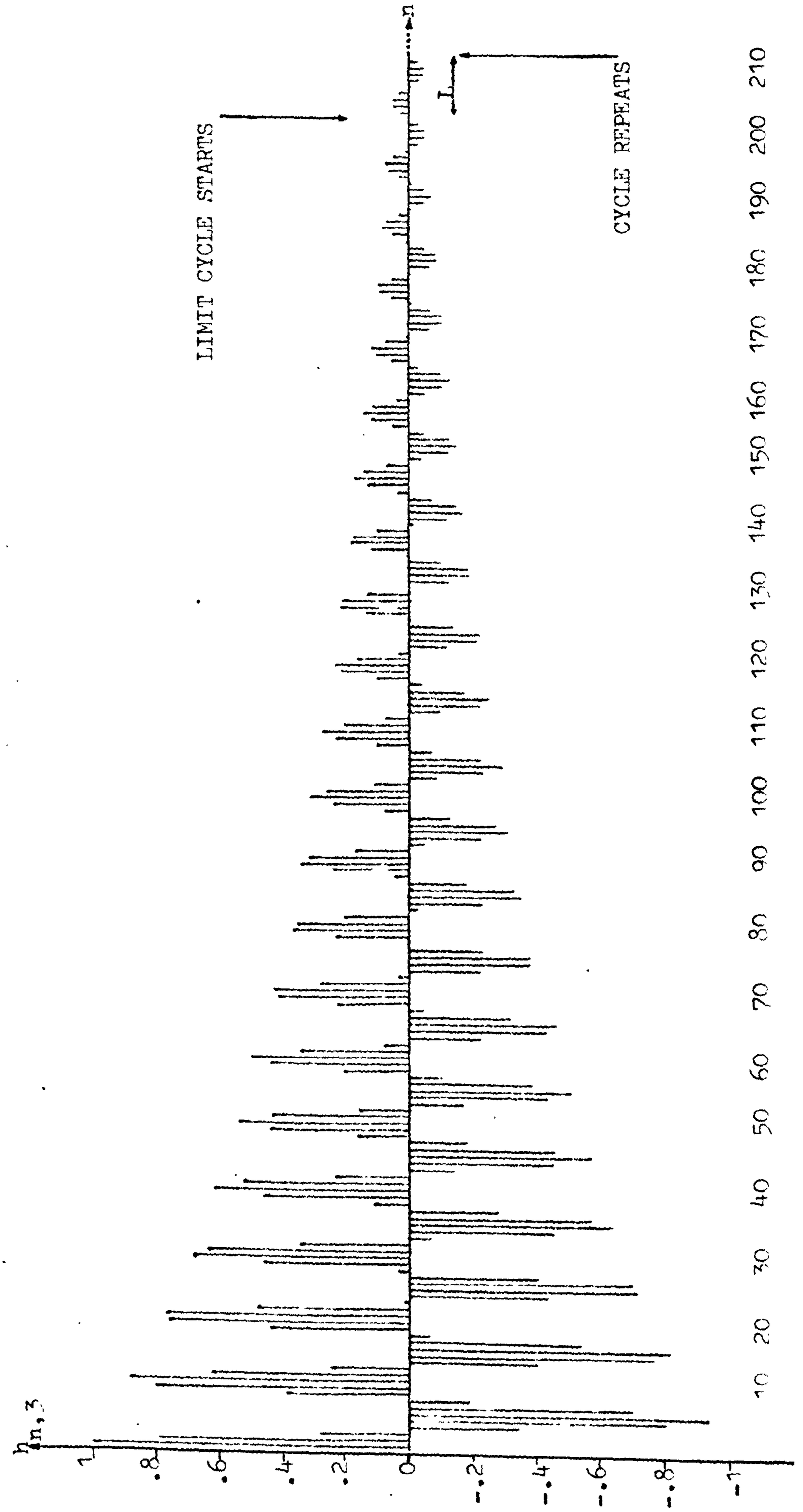


FIG: 8.2.13 EXPERIMENTAL IMPULSE RESPONSE OF THE SECOND SECOND-ORDER PASSBAND ELEMENTAL FILTER $H_2^P(z)$ OF BANDPASS FILTER C (WORDLENGTH: 8 BITS INCLUDING SIGN).

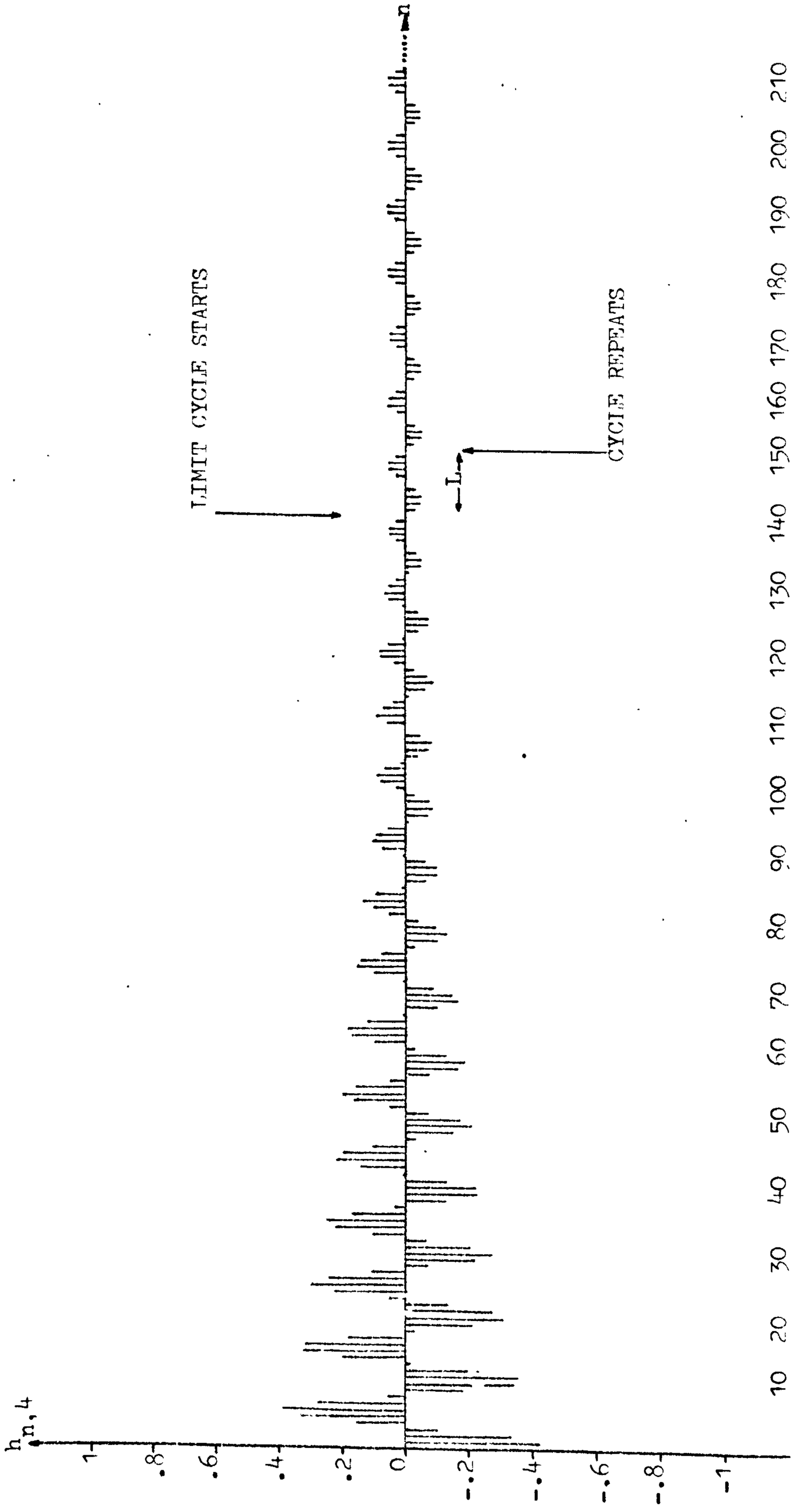


FIG: 8.2.14 EXPERIMENTAL IMPULSE RESPONSE OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER $-H_u^T(z)$ OF BANDPASS FILTER C (WORDLENGTH: 8 BITS INCLUDING SIGN).

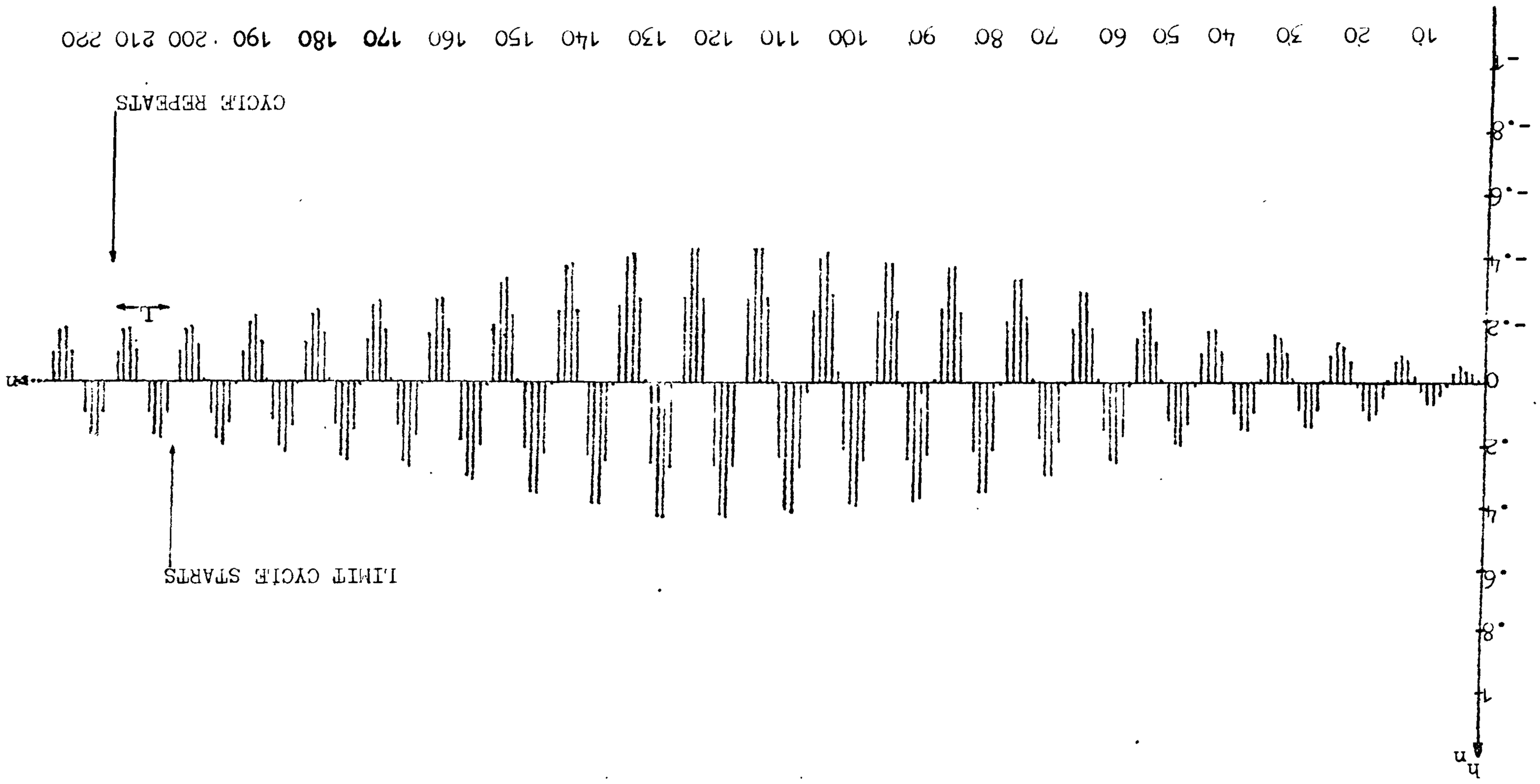


FIG: 8.2.15 EXPERIMENTAL IMPULSE RESPONSE OF HIGHER-ORDER BANDPASS FILTER C REALIZED IN CONFIGURATION (a) IN FIGURE 8.1.2 WITH **TRANSITION SAMPLE** W_n QUANTIZED TO A VALUE OF 0.414978027 (HORIZ ENGTH: 8 BITS INCLUDING SIGN).

ratio of its maximum amplitude to the dynamic range is 23/128= 0.1796875 or just under 18%. This is comparable to the maximum amplitude of the limit cycle of bandpass filter B, although the resultant limit cycle of bandpass filter C is approximately sinusoidal in nature, while that of bandpass filter B is more complex. On the other hand, the maximum amplitude of the limit cycle of bandpass filter A is only sixteen quantization steps or 12½% of the dynamic range. Hence, it is concluded that configuration (b) shown in figure 8.1.2 in the realization of a eighth-order bandpass filter design is preferred over configuration (a) shown in figure 8.1.2, especially when limit cycle behaviour is concerned, as explained previously. The above comments will apply to any given wordlength in general. In the next section, we shall consider the simulated impulse responses of bandpass filters A, B and C using word lengths longer than bits in representing the filter variables. This then gives an increased dynamic range with a correspondingly decreased width of the quantization step size.

8.2.2 SOME SIMULATED IMPULSE RESPONSES PERTINENT TO SECTION 8.2.1:

Simulations of the bandpass filters A, B and C, represented by equations (8.2.2), (8.2.3) and (8.2.4) respectively, are based on the simulation models shown in figures 8.1.3 and 8.1.4. Consider, for a start, the simulated impulse response of bandpass filter A using the simulation model in figure 8.1.4, with twelve bits including sign to represent the filter variables. Figure 8.2.16 shows the simulated impulse response of bandpass filter A with an input impulse value of 0.9995117188. A ringing with a maximum amplitude of 0.0471191, which corresponds to ninety-six quantization steps (the width of a quantization step is 0.0004882812502 in this instance), is observed to start at the 241th sample. Such a ringing response was not present in the 8-bit (including sign) impulse response of bandpass filter A shown in figure 8.2.5 due to a distortion of the ideal (infinite accuracy) impulse response by finite word length effects. The above ringing is due to the sharp cutoff of the optimized bandpass filter A (in the optimization process, bandpass filter A was optimized for the steepest cutoff) frequency response (in general, when filters in the Chebyshev or Butterworth classes are encountered, a ringing

response is further accentuated by a rising gain characteristic preceding the discontinuities of the cutoffs of their frequency responses). Nevertheless, such a ringing response will be shortened in a filter with a less sharp cutoff frequency response.

Furthermore, it is noticed from figure 8.2.16 that the above ringing dies down to a limit cycle at the 421st sample. The period of this limit cycle is L=10 such that it repeats every ten samples. The maximum amplitude of this limit cycle is nine quantization steps so that the ratio of its maximum amplitude to the dynamic range is 9/2048=0.00439453 or approximately 0.44%. Moreover, such a magnitude is indeed very small as compared with that shown in figure 8.2.5 previously and it may be considered that twelve bits including sign would be adequate to represent the filter variables of bandpass filter A. That this is indeed the case will further be substantizted by comparing figures 8.2.16 and 8.2.19. Finally, it is observed that the above limit cycle is approximately sinusoidal in nature and had there been no finite word length effects, the above ringing would have died down to zero value eventually as shown in figure 8.2.19.

Consider next the simulated impulse response of bandpass filter B using the simulation model in figure 8.1.3, with twelve bits including sign to represent the filter variables. Figure 8.2.17 shows the simulated impulse response of bandpass filter B with an input impulse value of 0.9995117188. The same ringing is observed as expected. The maximum amplitude of this ringing is ninety-five quantization steps or 0.0463867 which just about the same as that in the case of bandpass filter A. Furthermore, it is noticed from figure 8.2.17 that the above ringing dies down to a limit cycle at the 421st sample as in the case of bandpass filter A. The period of this limit cycle is L=10 such that it repeats every ten samples. The maximum amplitude of this limit cycle is, however, eleven quantization steps so that the ratio of its maximum amplitude to the dynamic range is 11/2048=0.00537109 or approximately 0.54%. This is as expected, since the realization configuration for bandpass filter B is less desirable as far as limit cycle behaviour is concerned as previously explained. Again, it is observed that the above limit cycle is approximately sinusoidal in nature and had there been no finite word

length effects, the above ringing would have died down to zero value eventually as shown in figure 8.2.20.

Finally, we consider the simulated impulse response of bandpass filter C using the simulation model in figure 8.1.3, with twelve bits including sign to represent the filter variables. Figure 8.2.18 shows the simulated impulse response of bandpass filter C with an input impulse value of 0.9995117188. A similar ringing is observed with a slightly decreased maximum amplitude of 0.0444336 or ninety-one quantization steps. The decrease in the amplitude of the above may be explained by the fact that the value of W_n is not optimized for the steepest cutoff in the frequency response of bandpass filter C; the value of W_n for bandpass filter C has been reduced to slightly less than the optimum to give a more gradual cutoff. Furthermore, it is noticed from figure 8.2.18 that the above ringing dies down to a limit cycle at the 421st sample with a period L=10 so that it repeats every ten samples. The maximum amplitude of this limit cycle is sixteen quantization steps so that the ratio of its maximum amplitude to the dynamic range is 16/2048=0.0078125 or approximately 0.8%. It is also observed that the above limit cycle is approximately sinusoidal in nature, and had there been no finite word length effects, the above ringing would have died down to zero value eventually as shown in figure 8.2.21. As in the case of bandpass filter A, ringings were not observed in figures 8.2.10 and 8.2.15 since the ideal impulse responses (shown in figures 8.2.20 and 8.2.21) for both cases were distorted by finite word length effects.

From the above impulse responses, it is observed that by increasing the word length used in representing the filter variables, the maximum amplitudes of the resultant limit cycles have accordingly descreased. In fact, such a result has been predicted by the analysis in section 6.8 in chapter six where the proposed generalized quantization noise model was used to analyze limit cycles implemented by the proposed CBFCS approach. Hence, the results in this and the previous sections have further confirmed the validity of the proposed noise model and in the next section we shall further analyze the above results on limit cycles. Nevertheless, to conclude this section, we shall now comment on the simulated impulse responses of bandpass filters A, B and C using an infinite word length for the variables.

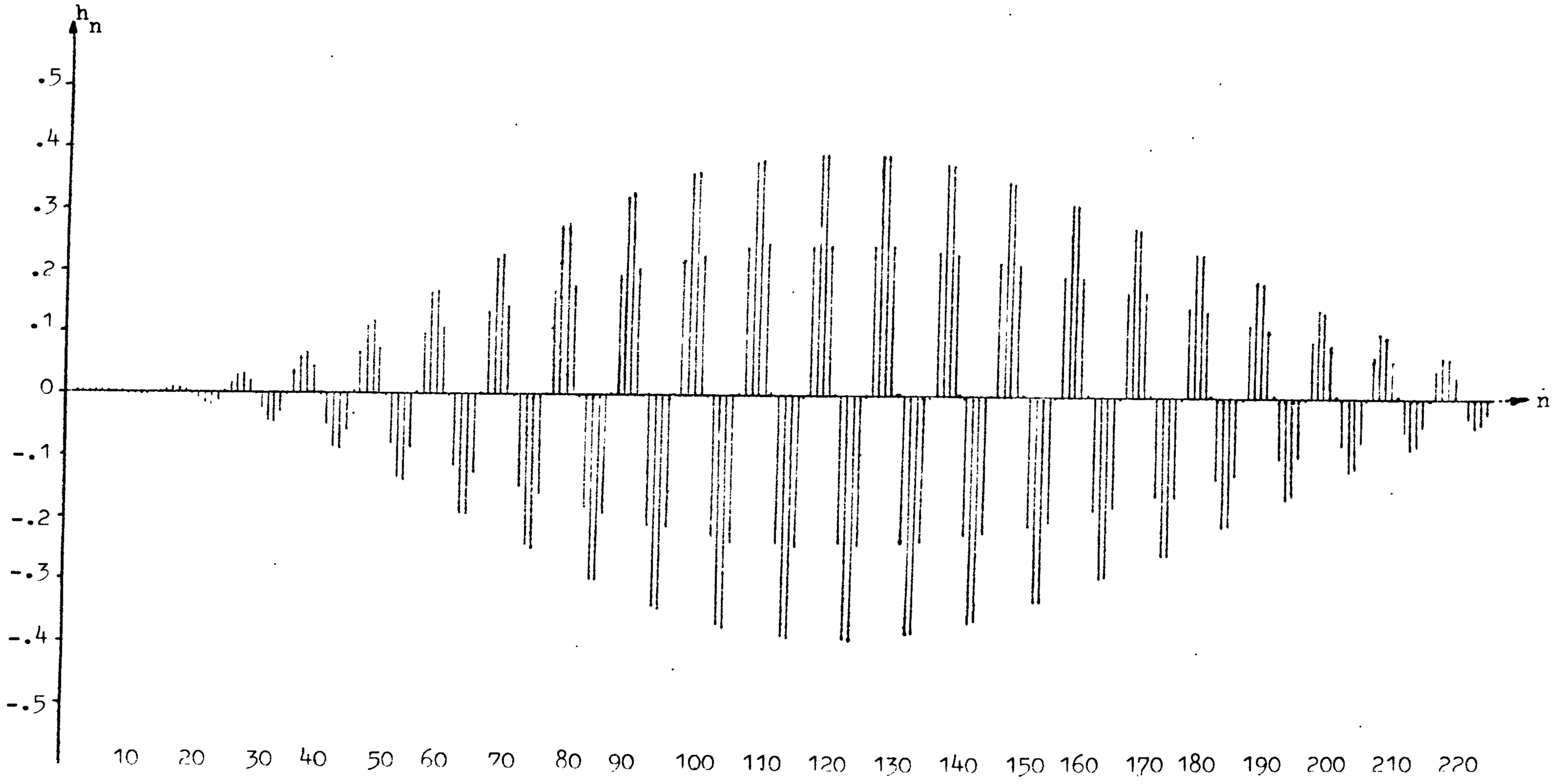
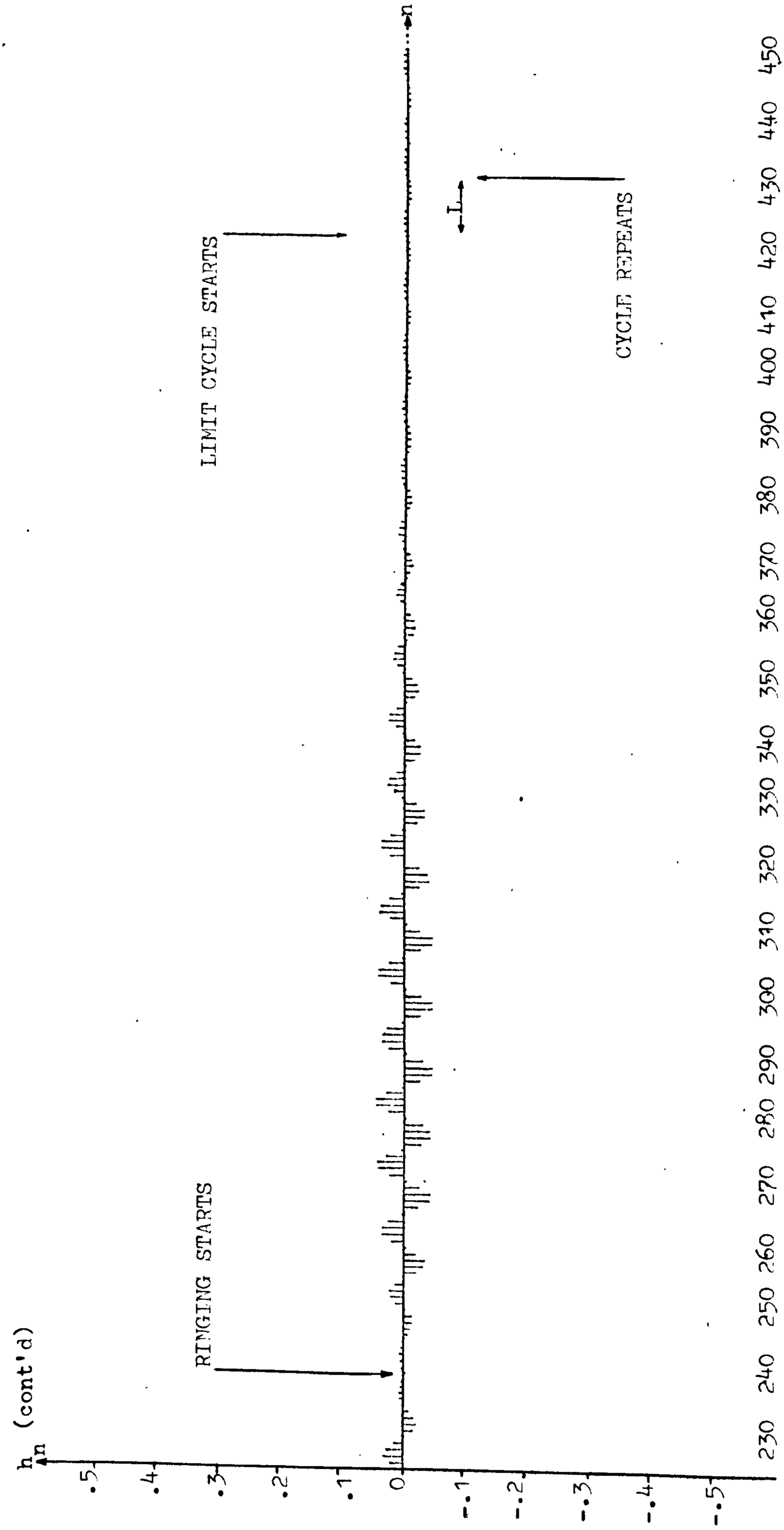


FIG: 8.2.16 SIMULATED IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER A REALIZED IN CONFIGURATION (b) IN FIGURE 8.1.2 WITH THE OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.5 (WORDLENGTH: 12 BITS INCLUDING SIGN).



CONTINUATION OF FIGURE 8.2.16

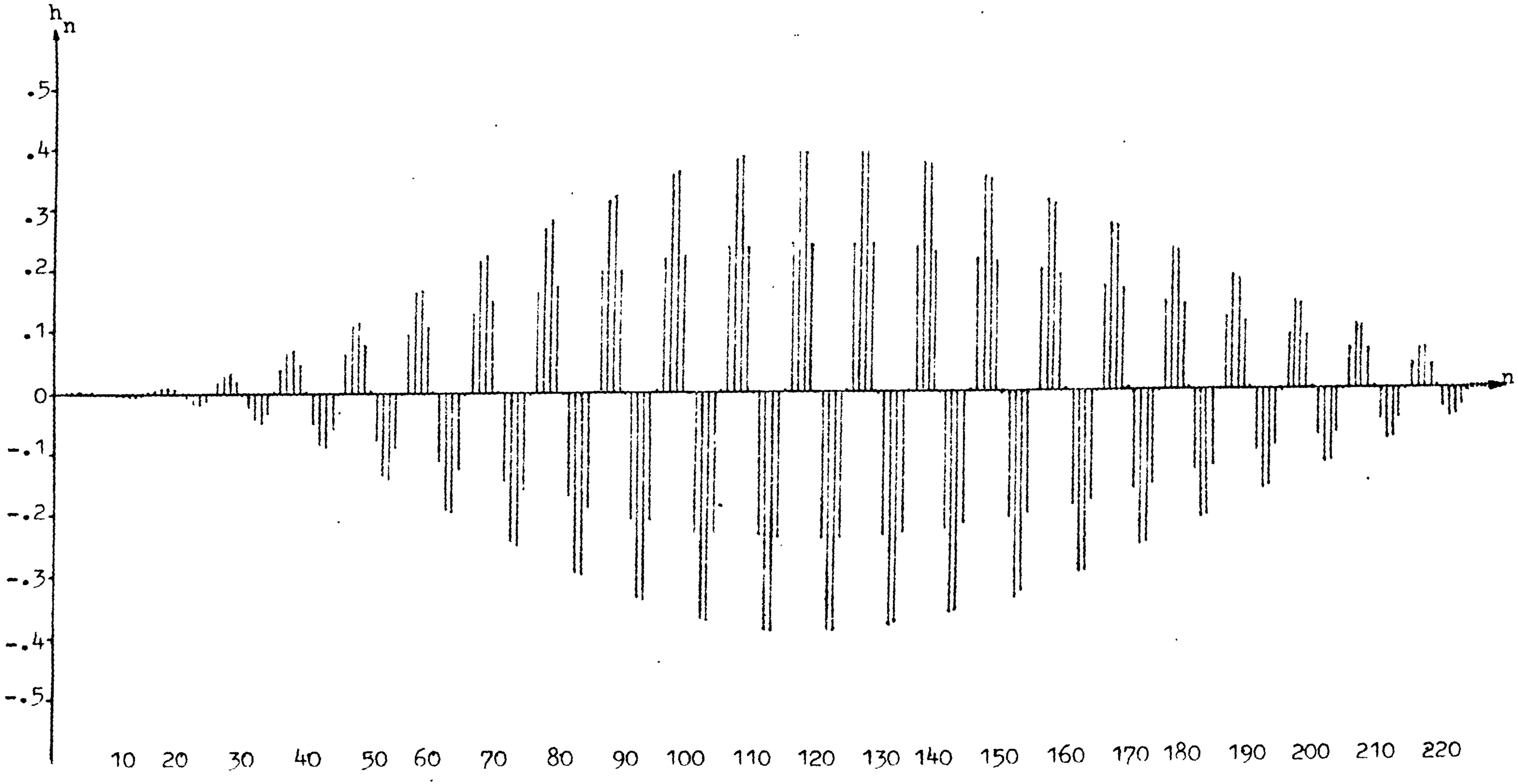
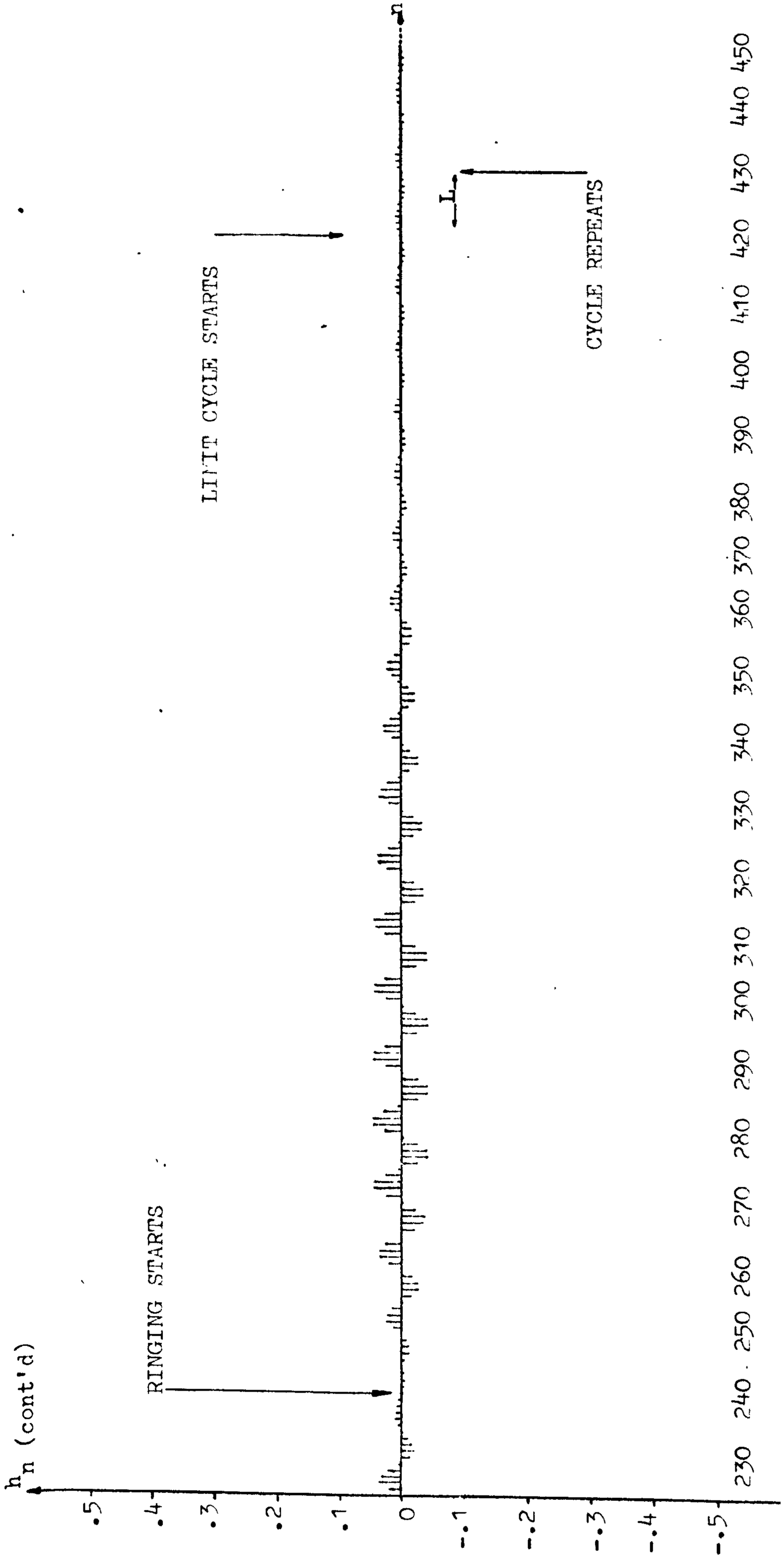


FIG: 8.2.17 SIMULATED IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER B REALIZED IN CONFIGURATION (a) IN FIGURE 8.1.2 WITH THE OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.5 (WORDLENGTH: 12 BITS INCLUDING SIGN).


h_n (cont'd)
RINGING STARTS
LIMIT CYCLE STARTS
n
L
CYCLE REPEATS
CONTINUATION OF FIGURE 8.2.17.

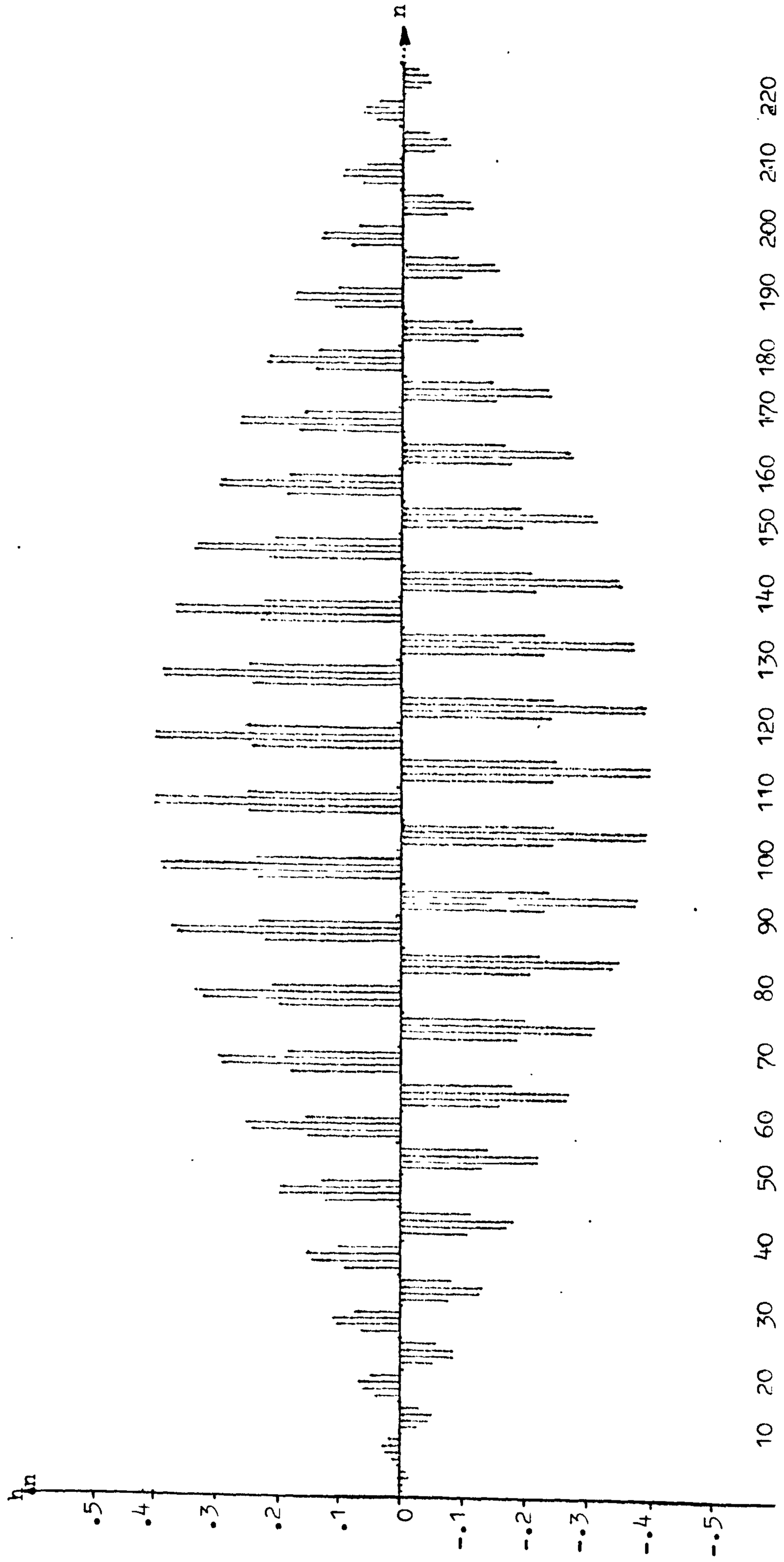
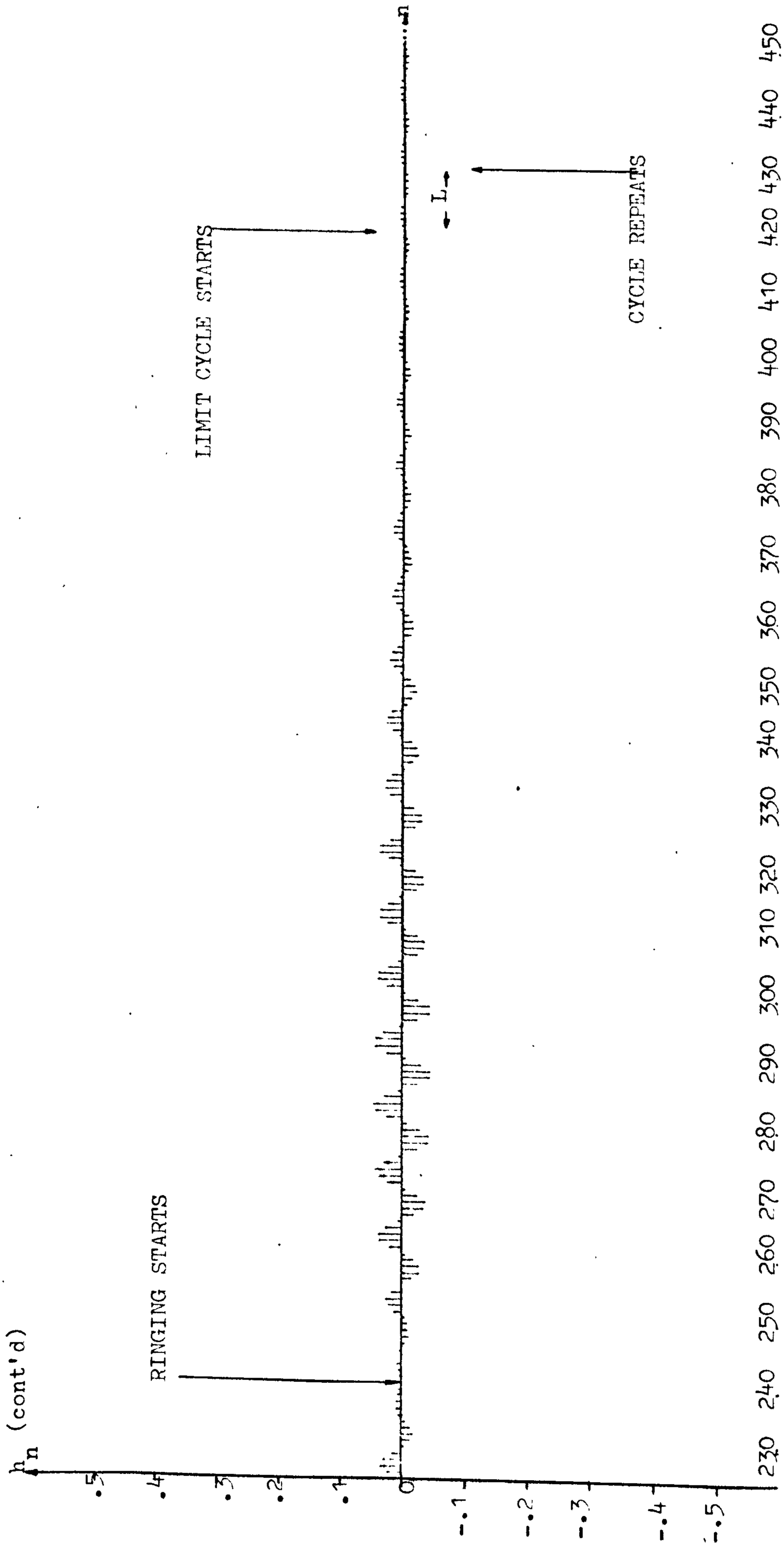


FIG: 8.2.18 SIMULATED IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER C REALIZED IN CONFIGURATION (a) IN FIGURE 8.1.2 WITH TRANSITION SAMPLE W_n QUANTIZED TC A VALUE OF 0.41497802777 (WORDLENGTH: 12 BITS INCLUDING SIGN).



COMBINATION OF FIGURE 8.2.18.

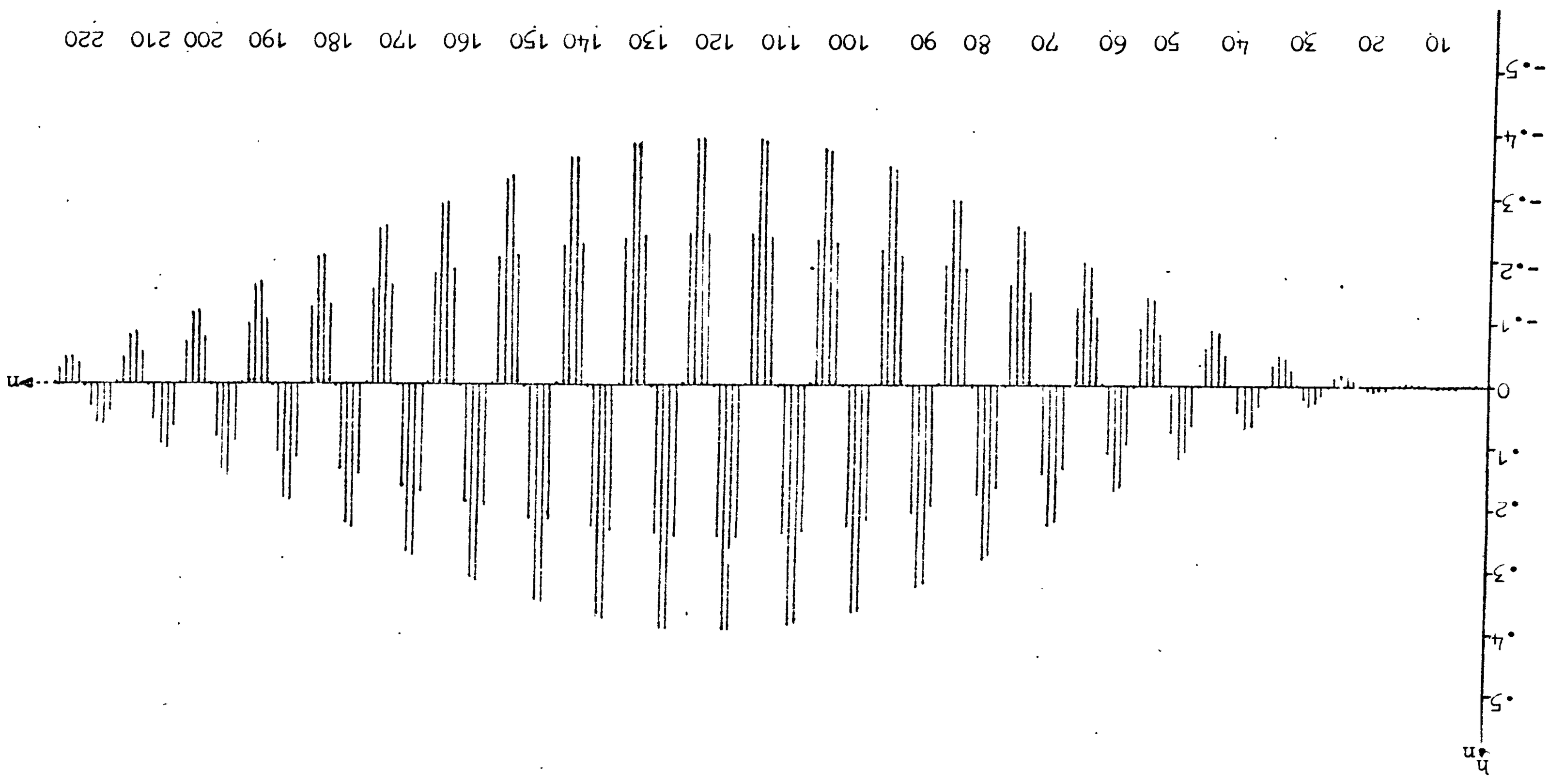
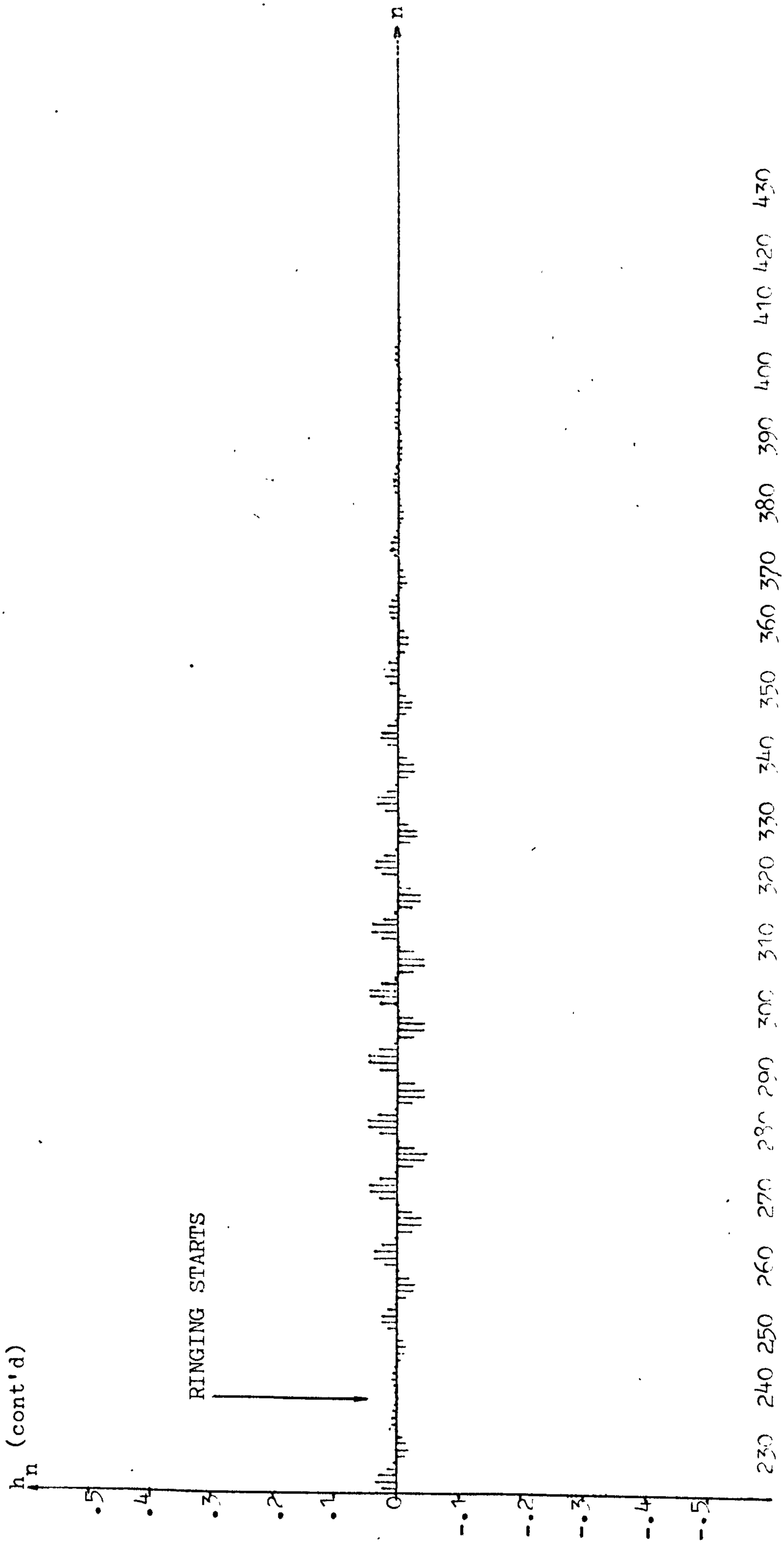


FIG: 8.2.19 SIMULATED IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER A REALIZED IN CONFIGURATION (b) IN FIGURE 8.1.2 WITH THE OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.5 (WORDLENGTH: INFINITE).



CONTINUATION OF FIGURE 8.2.19.

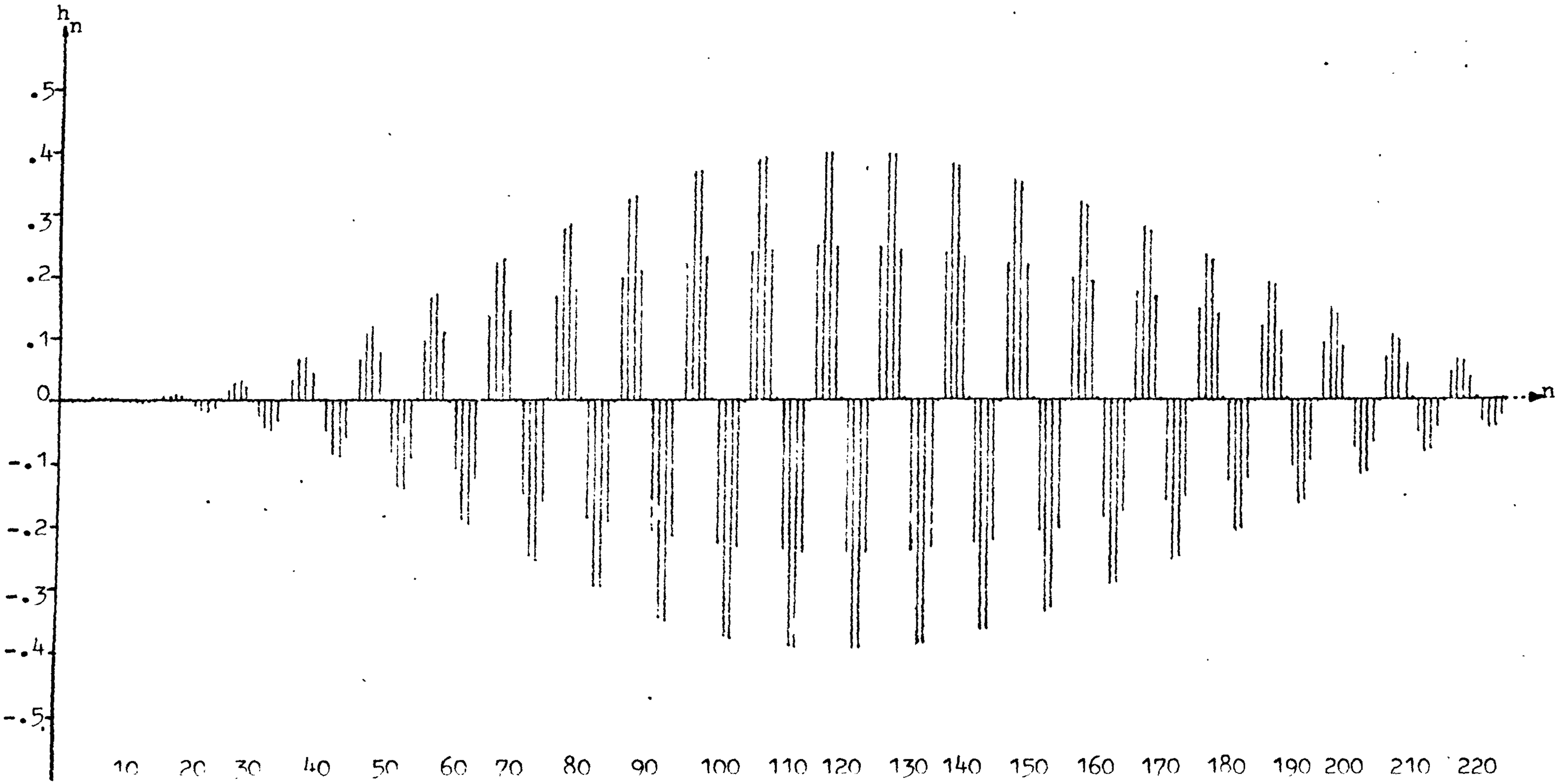
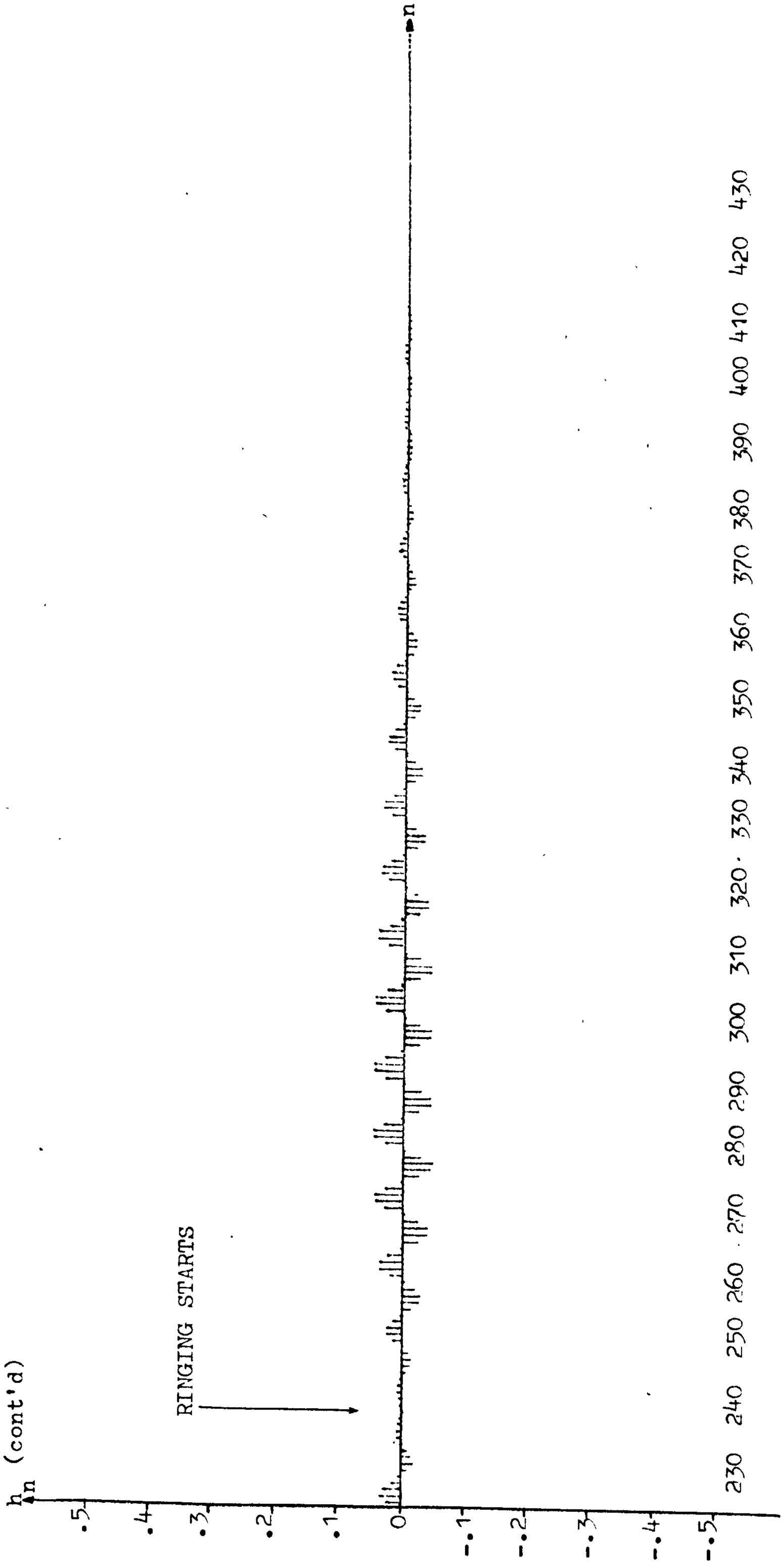


FIG: 8.2.20 SIMULATED IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER B REALIZED IN CONFIGURATION (a) IN FIGURE 8.1.2 WITH THE OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.5 (WORDLENGTH: INFINITE).



CONTINUATION OF FIGURE 8.2.20.

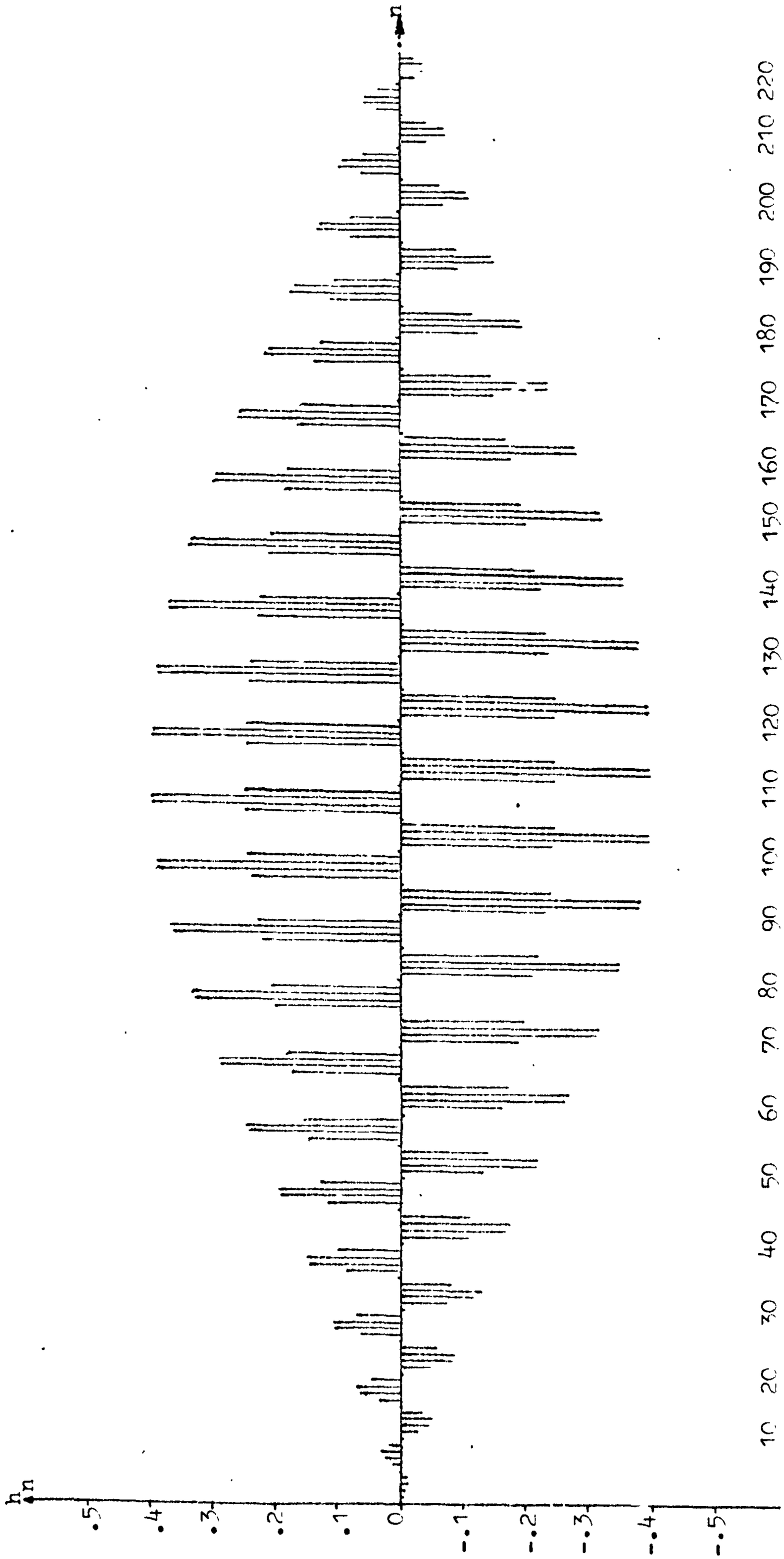
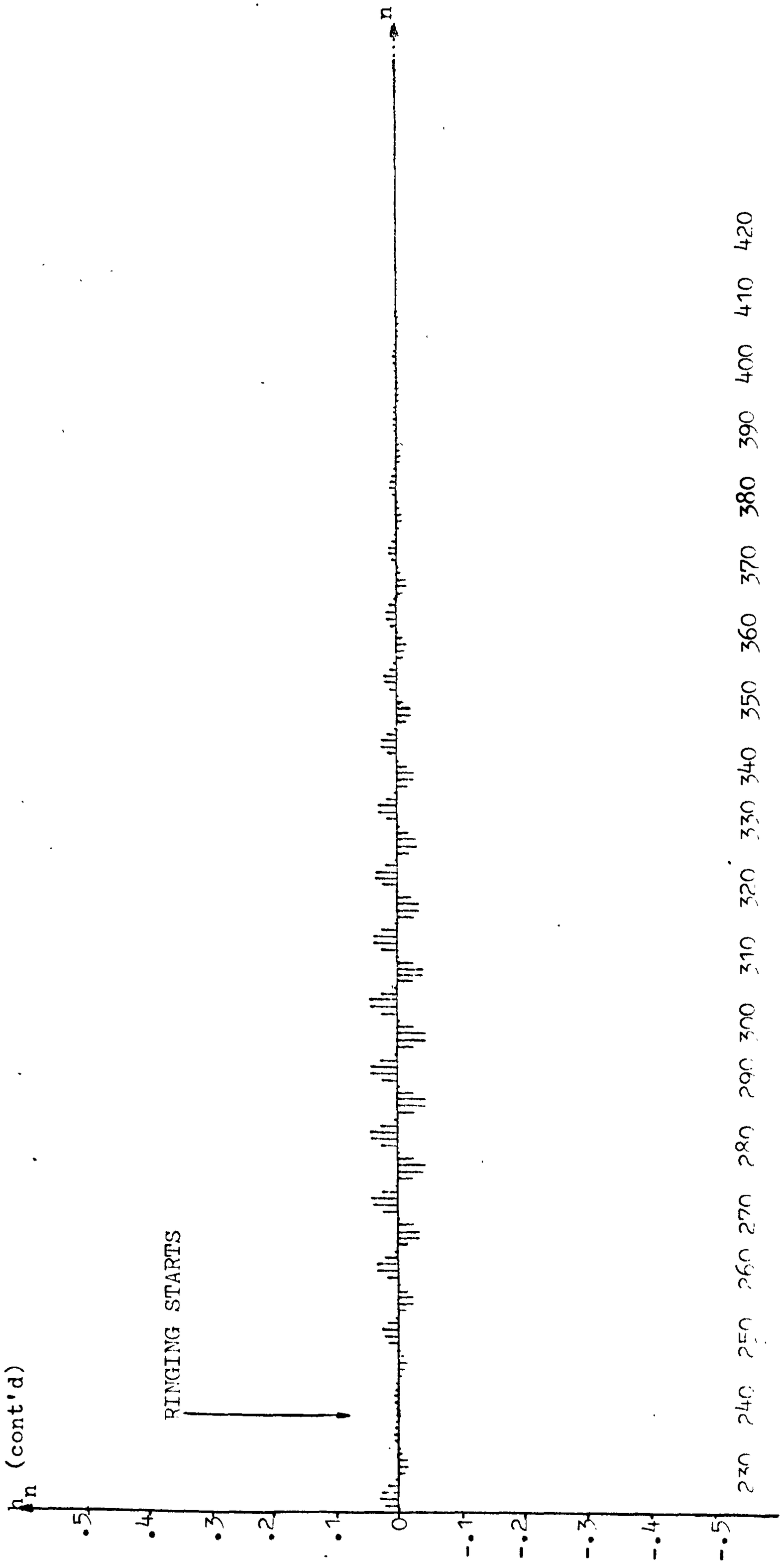


FIG: 8.2.21 SIMULATED IMPULSE RESPONSE OF EIGHTH-ORDER BANDPASS FILTER C REALIZED IN CONFIGURATION (a)
IN FIGURE 8.1.2 WITH TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.4149780277
(SELECTIVE INTENSITY : HIGHEST)



CONTINUATION OF FIGURE 8.2.21.

Figures 8.2.19, 8.2.20 and 8.2.21 show the simulated impulse responses of bandpass filters A, B and C using an infinite word length. A comparison shows that the ringing in the infinite word length impulse response of bandpass filter C shown in figure 8.2.21 is fractionally shorter and also smaller in amplitude. This has been mentioned before to be due to the fact that the transition sample W_n of bandpass filter C is non-optimum in the sense that the frequency response of the filter exhibits a less sharp cutoff characteristic than those of filters A and B.

Except for the slight ringing effects, it is further observed that there is an anti-symmetry associated in the above infinite word length impulse responses. It has been proved in reference (10) that a filter with an anti-symmetric impulse response will exhibit a piecewise linear phase response. Hence, the above ringing effects are seen to slightly distort the otherwise piecewise linear phase characteristics of the bandpass filters A, B and C designed by the proposed design technique discussed in chapter three previously (which has been pointed out to be a linear-phase approximation technique). The above anti-symmetry is also observed in figures 8.2.5, 8.2.10, 8.2.15, 8.2.16, 8.2.17 and 8.2.18 respectively. In these instances, the above anti-symmetry in these impulse responses has been spoilt by the resultant limit cycles due to finite word length effects.

In general, the above ringing effect can be suppressed by introducing a wider transition band(s) which implies a more gradual cutoff in the frequency response and a more linear phase response. Depending on the particular application, a designer can always make judicious comprises to suit the circumstances. Nevertheless, the high degree of phase-linearity coupled with reasonably sharp cutoff characteristics of filters designed by the proposed design technique will be shown in the latter part of the present chapter.

8.2.3 LIMIT CYCLE ANALYSIS:

In this section, we shall analyze the above results on limit cycles with respect to the bounds derived in section 6.8 in chapter six previously. Although equation (6.8.15) provides an absolute bound on the peak value of any limit cycle with period L

of a basic second-order section implemented by the proposed CBFCS algorithm, in general, the limit cycle periods for different initial conditions cannot be determined except by exact simulation. Thus, for design purposes, equation (6.8.16) then gives a bound which is independent on L, and can be written as follows:

$$|y'_n| \leqslant 2^{-M} \cdot \sum_{k=0}^{\infty} |h_k|$$

where in the limit L approaches infinity, equations (6.8.15) and (6.8.16) become identical as shown in equation (6.8.17). In the following we shall apply equation (6.8.16) to the experimental results in the previous section. Since the individual second-order elemental filters of bandpass filters A, B and C have complex poles, the summation of equation (6.8.16) cannot be put in closed form in general and the least upper bound on the absolute value of y'_n in equation (6.8.15) is given by

$$|y'_n| \leqslant 2^{-M} \cdot \sum_{k=0}^{\infty} |h_k| = 2^{-M} \cdot \sum_{k=0}^{\infty} \left| \frac{R^k}{\sin \omega_r T} \cdot \sin(\overline{k+1}\,\omega_r T) \right| \qquad (8.2.5)$$

where the general form of the above elemental filters is given by equation (6.4.1) with an expression for their impulse responses given by equation (B10) in appendix B. Useful estimates which approximate equation (8.2.5) are given below for design purposes:

$$|y'_n| \leqslant 2^{-M} \cdot \sum_{k=0}^{\infty} |h_k| \leqslant 2^{-M} \cdot \sum_{k=0}^{\infty} \left| \frac{R^k}{\sin \omega_r T} \right| = \frac{(1+R)}{(1-R^2)\sqrt{1-\frac{(2R\cos \omega_r T)^2}{4R}}} \qquad (8.2.6)$$

or

$$|y'_n| \leqslant 2^{-M} \cdot \sum_{k=0}^{\infty} |h_k| \leqslant 2^{-M} \cdot \sum_{k=0}^{\infty} (k+1) R^k = \frac{1}{(1-R)^2} \qquad (8.2.7)$$

for a general second-order elemental filter resonating at ω_r.

From the approximation made in equation (6.8.16) and the further approximations made in the above estimates, equations (8.2.6) and (8.2.7) may therefore represent pessimistic upper bounds on the absolute value of y'_n in equation (6.8.15), where the symbol y'_n has the usual interpretation as given in section 6.8 in chapter six. A tighter upper bound than equation (8.2.6) can be given as below:

$$|y'_n| \leq \frac{2^{-M}}{(1-R^2)\sqrt{1-\left(\frac{2R\cos\omega_c T}{4R}\right)^2}} \qquad (8.2.8)$$

which differs from equation (8.2.7) by a factor of (1+R).

Evaluating equation (8.2.6) for the elemental filters of bandpass filter A, we find that the above bound varies from four to twenty-one times as large as the exact maximum amplitudes of the limit cycles of the individual elemental filters. On the other hand, evaluating equation (8.2.8) for the elemental filters of bandpass filter A, it is found that the above bound varies from two to ten times as large as the exact maximum amplitudes of the limit cycles of the respective elemental filters. Finally, the bound given by equation (8.2.7) is overly pessimistic.

Evaluating equation (8.2.6) for the elemental filters of bandpass filter B, we find that the above bound varies from four to ten times as large as the exact maximum amplitudes of the limit cycles of the respective elemental filters. On the other hand, evaluating equation (8.2.8) for the same set of elemental filters, it is found that the above bound varies from two to five times as large as the exact maximum amplitudes of the limit cycles of the respective elemental filters. Again, the bound given by equation (8.2.7) is overly pessimistic.

Finally, evaluating equations (8.2.6),(8.2.7), (8.2.8) for the elemental filters of bandpass filter C, the same results as in the case of bandpass filter B have been obtained. This is as expected since both bandpass filters B and C were realized with the same hardware configuration (a) shown in figure 8.1.2. Furthermore, the limit cycle analysis for the limit cycles of the bandpass filters A

B and C discussed in the previous sections are obvious from the results obtained above from the analyses of the respective elemental filters.

8.2.4 FURTHER EXPERIMENTAL IMPULSE RESPONSES:

In this section we present further experimental impulse responses obtained with the demonstration processor. Consider the following design with desired specifications:

(a) A bandpass filter with a nominal bandwidth of 100Hz.
(b) Centre frequency at 1MHz, i.e. percentage bandwidth is 0.01%.
(c) Passband ripple less than 0.2dB.
(d) The transition bands on both sides of the passband should be as narrow as possible, that is, a steep transition skirt.

In order to meet the above specifications, an eighth-order filter was used, with two passband poles and a pair of transition poles, one on each side of the passband. Furthermore, this 4-pole bandpass filter was realized with four second-order elemental filters centred at 999950Hz, 999975Hz, 1000025Hz and 1000050Hz respectively to give a nominal bandwidth of 100Hz. Thus, the passband poles are seperated by 50Hz in frequency while the transition poles are each seperated from its nearest passband pole by 25Hz in frequency. This then give the value of b in equation (8.1.1) to be 50π radians. With this value of b, the value of a in equation (8.1.1) was further chosen to be 50π radians for large out-of-band rejection and steep transition. Hence, the 3dB bandwidth of individual elemental filters is given to be 50Hz by equation (3.7.1), and the percentage bandwidth of an elemental filter is therefore 0.005%. Furthermore, for such a percentage bandwidth of an elemental filter and the desired specification given in (b) above, a sampling frequency as low as three times the centre frequency, i.e. 3MHz, was found to be adequate, without introducing any significant frequency aliasing effects. (This gives the ratio BT as 1:60000 which indicates that any aliasing effect present will be negligible).

The resultant transfer function was then optimized according to the optimization procedure discussed in section 3.2.6 to yield the steepest transition skirt possible. The theoretical magnitude of the

passband ripple before optimization is given by equation (3.2.21) as 0.38 dB approximately. However, after optimzation, the value was found to be 0.13dB approximately. The value of the optimized W_n was found to be 0.46. Substituting these values into equation (8.1.1), the resultant transfer function H(z) is written as:

$$H(z) = \frac{0.46(1 + 0.4998831327z^{-1})}{1 + 0.9997662654z^{-1} + 0.9998952857z^{-2}}$$

$$- \frac{(1 + 0.4999284777z^{-1})}{1 + 0.9998569554z^{-1} + 0.9998952857z^{-2}}$$

$$+ \frac{(1 + 0.5000191629z^{-1})}{1 + 1.0000038326z^{-1} + 0.9998952857z^{-2}}$$

$$- \frac{0.46(1 + 0.5000645025z^{-1})}{1 + 1.0001290005z^{-1} + 0.9998952857z^{-2}} \qquad (8.2.9)$$

which was finally quantized as described in section 6.4.2 to give $H_Q(z)$ with coefficients represented to 16-bit accuracy including sign. The quantized transfer function is therefore written as:

$$H_Q(z) = \frac{0.453125(1 + 0.4998779301z^{-1})}{1 + 0.9997557719z^{-1} + 0.9999084943z^{-2}}$$

$$- \frac{(1 + 0.49992366z^{-1})}{1 + 0.9998473199z^{-1} + 0.9999084943z^{-2}}$$

$$+ \frac{(1 + 0.500030517578125z^{-1})}{1 + 1.000030517578125z^{-1} + 0.9999084943z^{-2}}$$

$$- \frac{0.453125(1 + 0.500061035156250z^{-1})}{1 + 1.000122070312500z^{-1} + 0.9999084943z^{-2}} \qquad (8.2.10)$$

where $H_Q(z)$ was realized in configuration (b) shown in figure 8.1.2 with the value of W_n re-optimized and quantized to 0.453125.

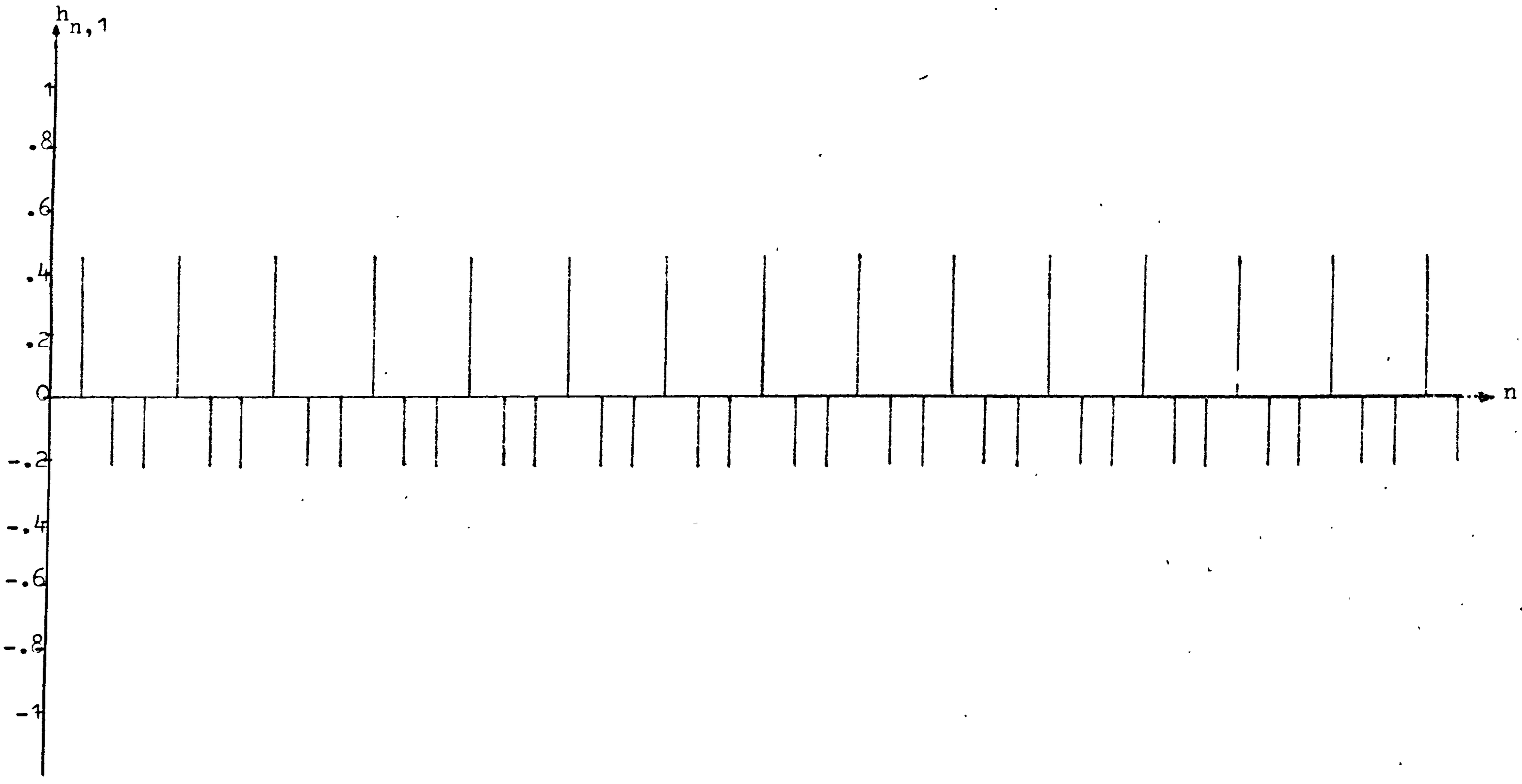


FIG: 8.2.22 EXPERIMENTAL IMPULSE RESPONSE OF THE LOWER TRANSITION SECOND-ORDER ELEMENTAL FILTER $H_1^T(z)$ OF A BANDPASS FILTER CENTRED AT 1MHz WITH OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.453125 (WORD LENGTH: 8 BITS INCLUDING SIGN).

FIG: 8.7.23 EXPERIMENTAL IMPULSE RESPONSES OF THE FIRST SECOND-ORDER PASSBAND SEGMENTAL FILTER $-H_1^P(z)$ OF A BANDPASS FILTER CENTERED AT 1GHz (WORD LENGTH: 8 BITS INCLUDING SIGN).

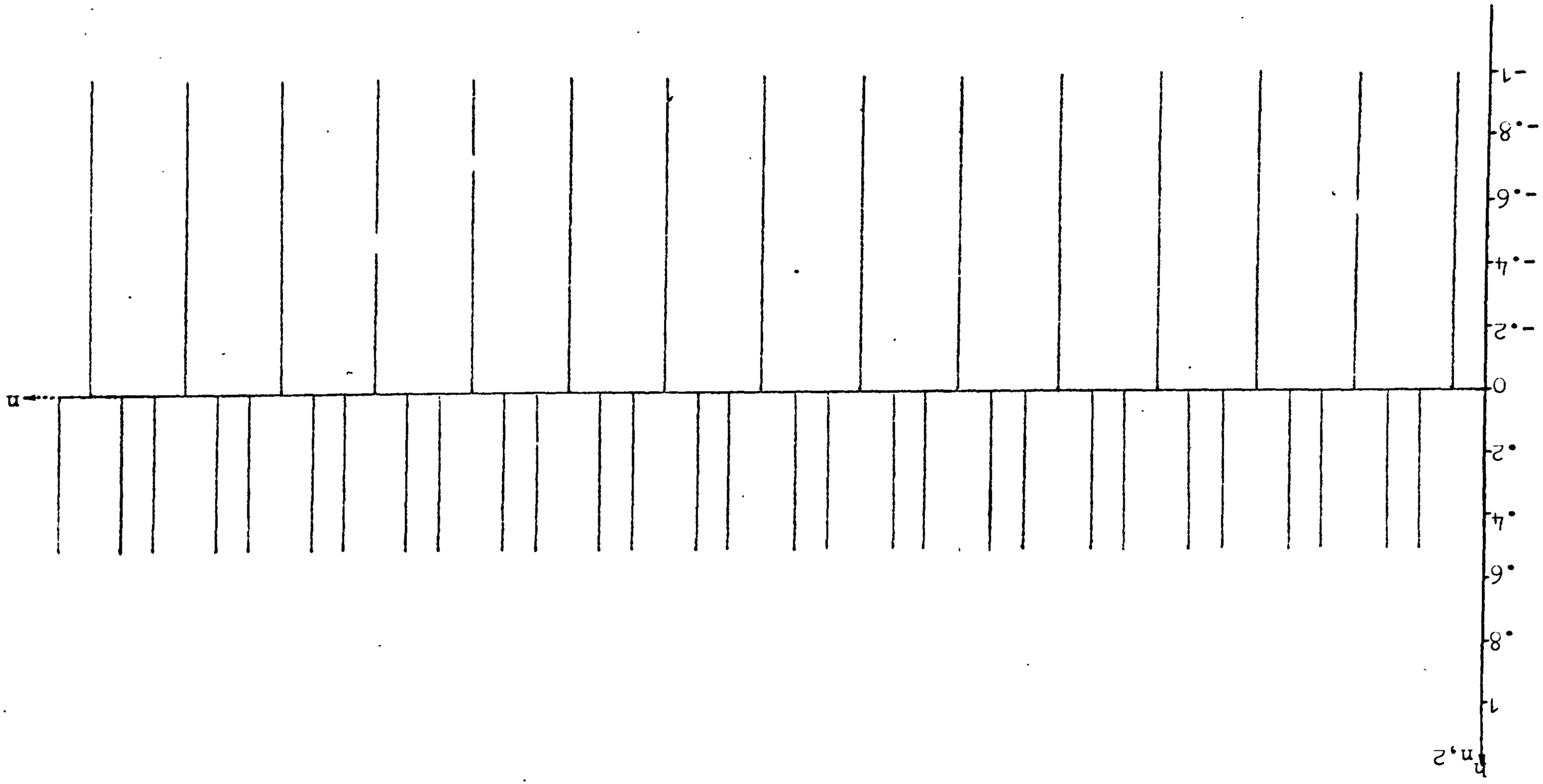


FIG: 8.2.24 EXPERIMENTAL IMPULSE RESPONSE OF THE SECOND-ORDER PASSBAND ELEMENTAL FILTER $H_2^P(z)$ OF A BANDPASS FILTER CENTERED AT 1MHz (VHF) LENGTH: 8 BITS INCLUDING SIGN (NDS ONLY SETS)

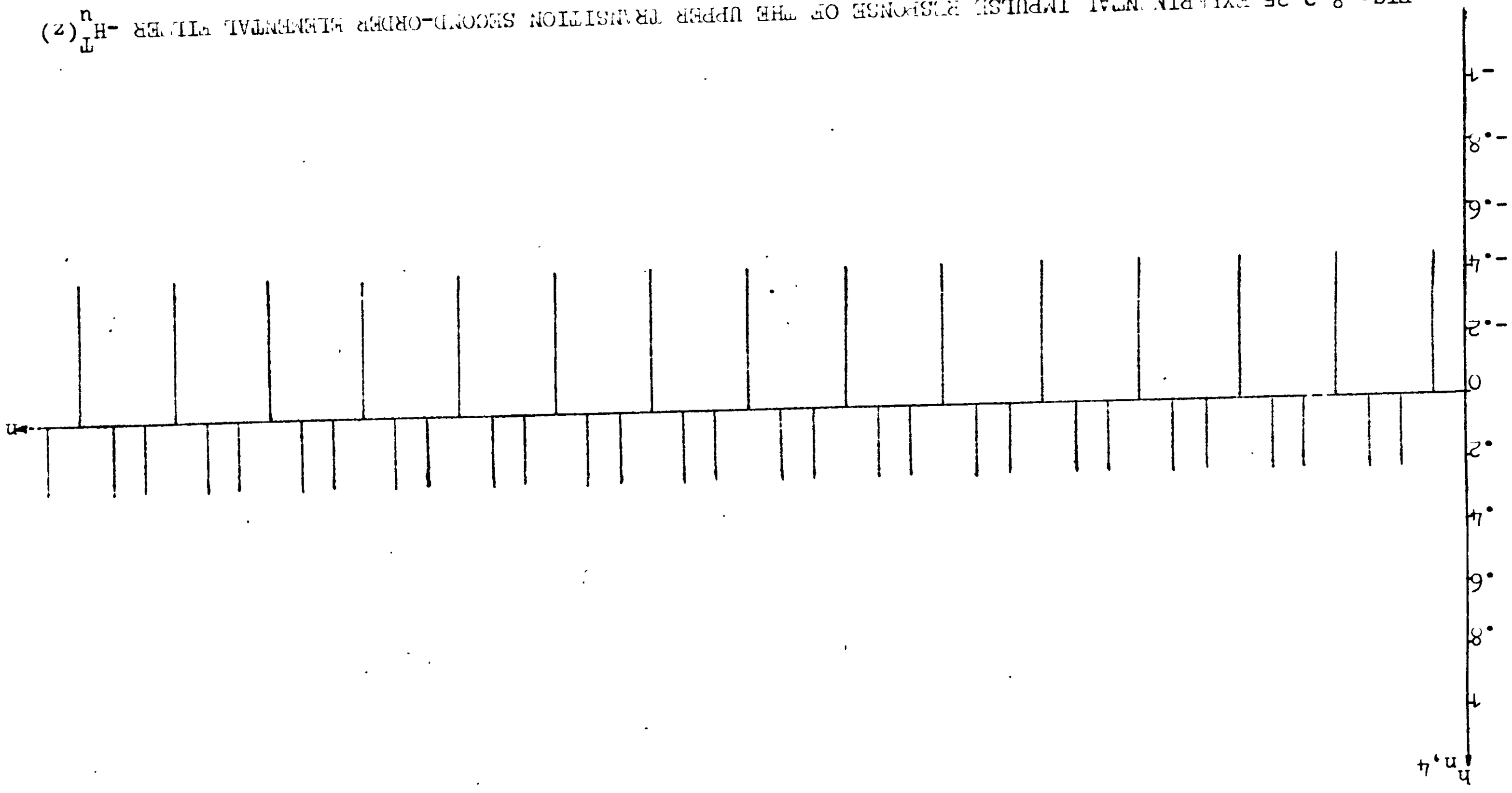


FIG: 8.2.25 EXHIBITING THE IMPULSE RESPONSE OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER $-H_u^T(z)$ OF A PANDPASS FILTER CENTERED AT 1MHz WITH OPTIMIZED TRANSITION SAMPLE W_n QUANTIZED TO A VALUE OF 0.452125 (7 BINARY DIGITS: 8 BITS INCLUDING SIGN).

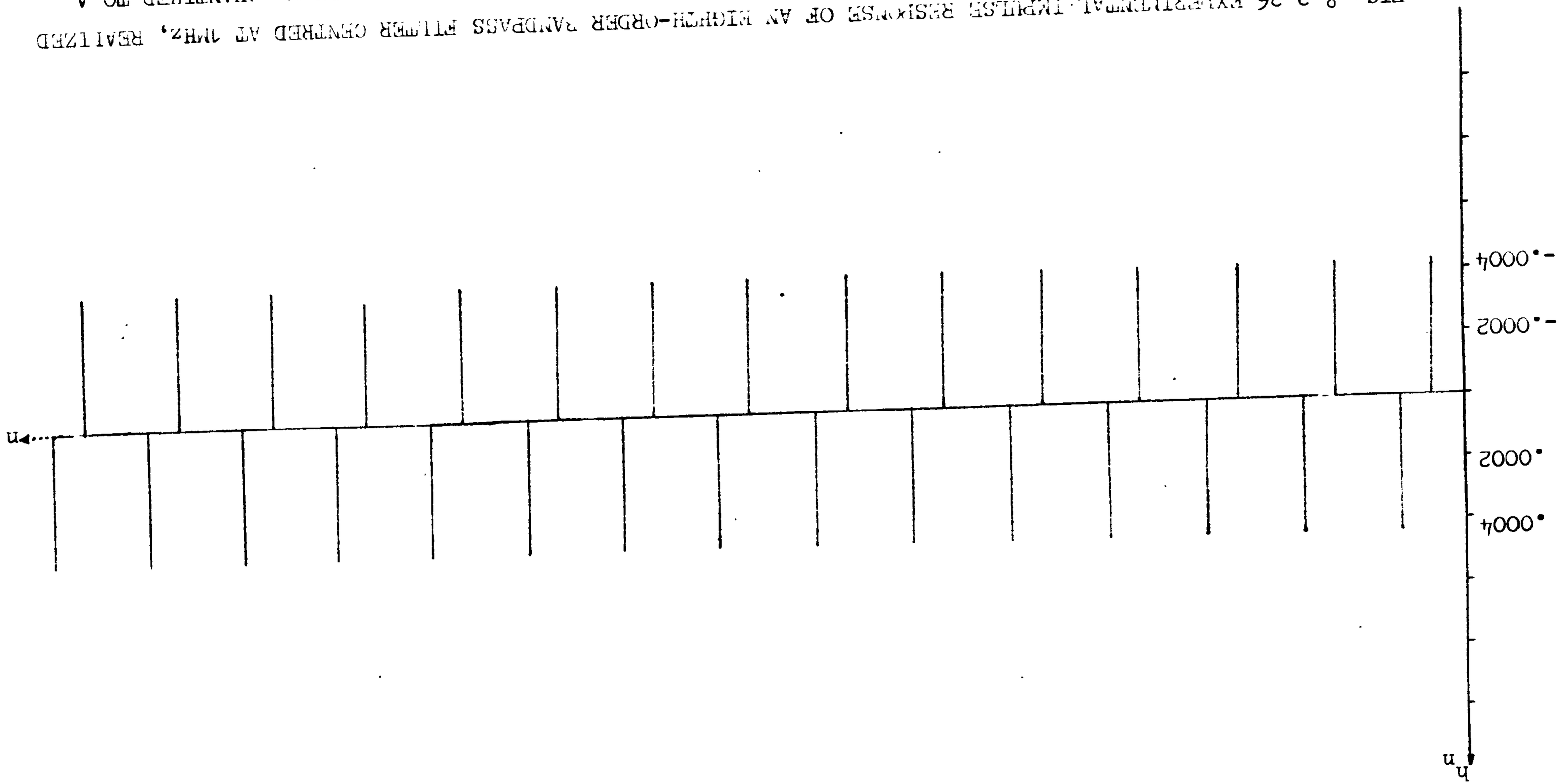


FIG: 8.2.26 EXPERIMENTAL IMPULSE RESPONSE OF AN EIGHTH-ORDER BANDPASS FILTER CENTERED AT 1MHz, REALIZED IN CONFIGURATION (b) IN FIGURE 8.1.2 WITH OPTIMIZED TRANSITION SAMPLE V_n QUANTIZED TO A VALUE OF 0.453125 (2's COMPLEMENT: 8 BITS INCLUDING SIGN).

Figures 8.2.22 to 8.2.26 show the experimental impulse responses of the four second-order elemental filters and that of that of the resultant eighth-order bandpass filter centred at 1MHz. The value of the unit impulse input was 0.1111111=0.9921875, while the respective contents of the CBFCS memories for the four second-order elemental filters are shown in tables C13 to C16 in appendix C. Due to finite word length effects, the responses of the four second-order elemental filters shown in figures 8.2.22 to 8.2.25 are seen to be constant-amplitude oscillations. These oscillations can be regarded as large-amplitude limit cycles of period three. Had there been no finite word length effects, one would expect the impulse responses of the four elemental filters to die down to zero value with the envelope of a decaying sinusoid. Although no overflows occur, the above experimental impulse responses no longer resemble the ideal (infinite accuracy) responses). Furthermore, the above oscillations are seen to be approximately sinusoidal in nature. Figure 8.2.26 shows the highly distorted impulse response of the 1MHz bandpass filter with a word length of eight bits including sign to represent its filter variables. The amplitude of the constant-amplitude oscillation shown in figure 8.2.26 is less than 0.0004 in peak value such that the resultant signal energy that passes through the above bandpass filter is minimal. In fact, this impulse response indicates that the above 1MHz bandpass filter, implemented with a word length of eight bits including sign, hardly represents a filter at all.

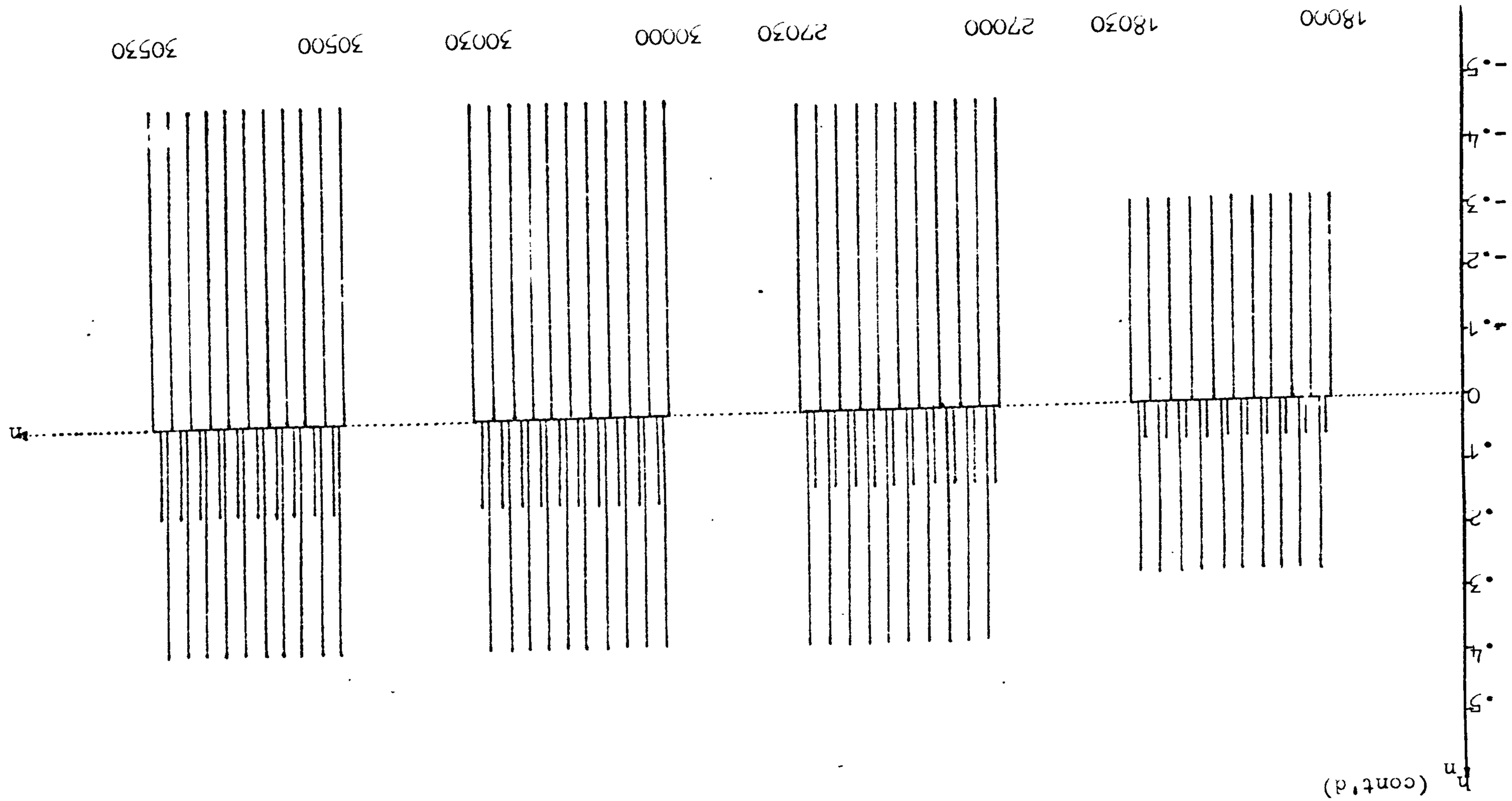
Hence, in view of the above, it is concluded that a word length of eight bits including sign is not sufficient to implement the above eighth-order bandpass filter centred at 1MHz. The simulated impulse response of the above bandpass filter using infinite accuracy will be shown in the next section.

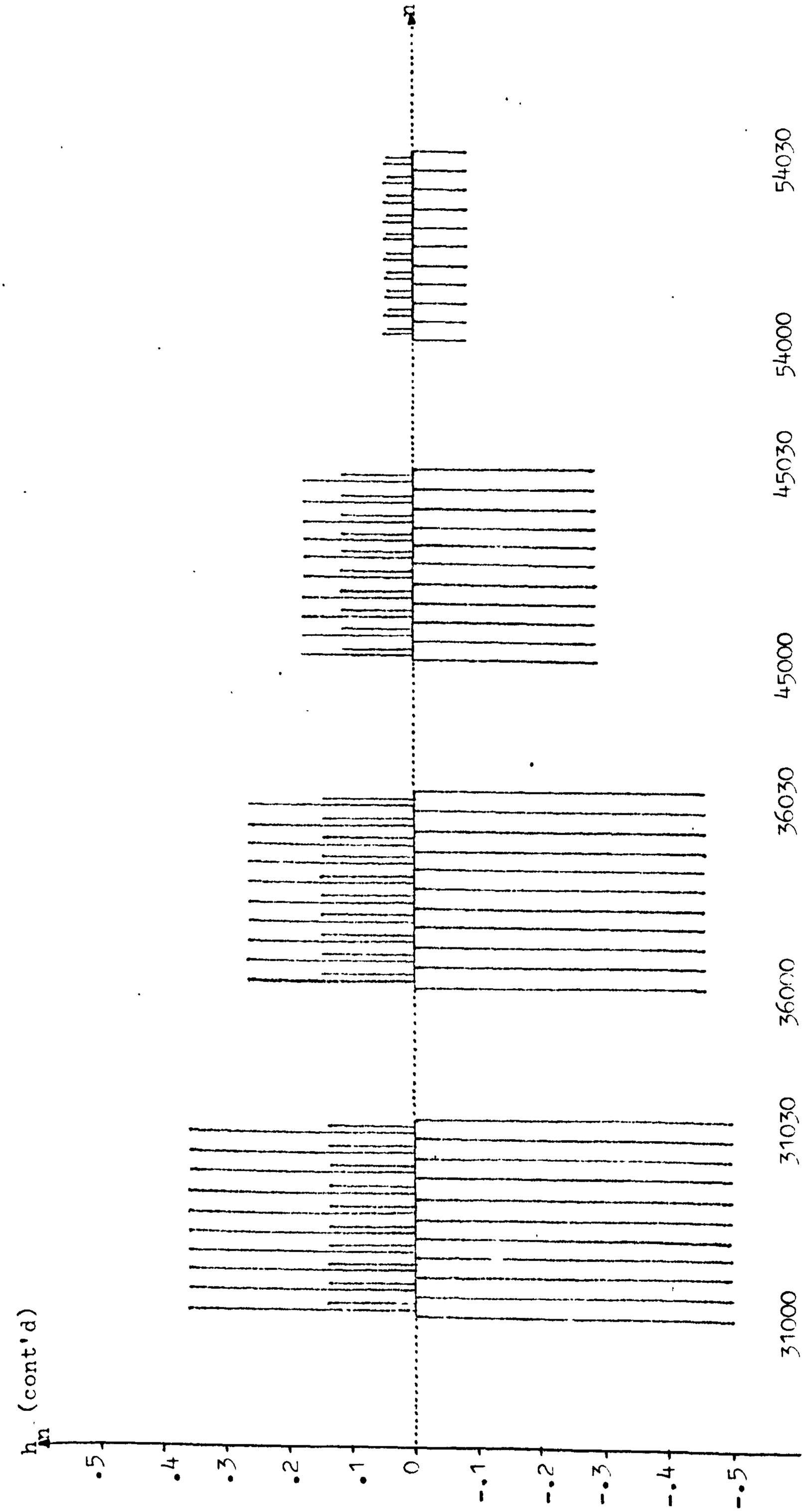
8.2.5 SIMULATED IMPULSE RESPONSE PERTINENT TO SECTION 8.2.4:

Simulations have been carried out for the above bandpass filter centred at 1MHz, using the simulation model shown in figure 8.1.4. Results show that a word length of over sixteen bits has to be used to realize a usable filter represented by equation 8.2.10 with limit cycles whose amplitudes are tolerable. Furthermore, figure 8.2.27

FIG: 8.2.27 STIMULATED IMPULSE RESPONSE OF AN EIGHTH-ORDER BANDPASS FILTER CENTRED AT 1MHz, REALIZED IN CONFIGURATION (b) IN FIGURE 8.1.2 WITH OPTIMIZED TRANSITION SAMPLE VALUE OF 0.453125 (ORIGINAL: INFINITE WORDLENGTH; EQUIVALENT FINITE WORDLENGTH: W_n QUANTIZED TO A

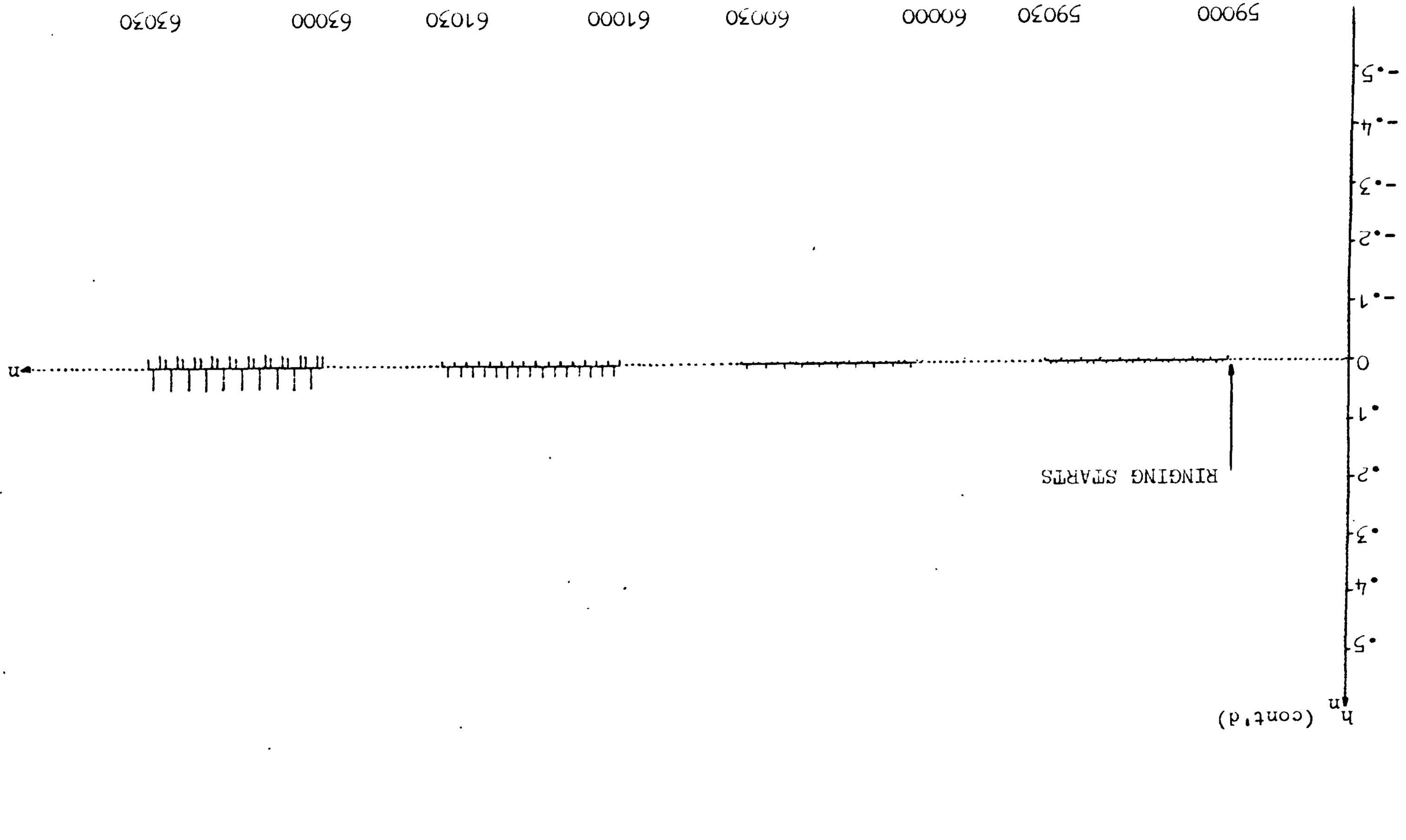
CONTINUATION OF FIGURE 8.2.27 SHOWING THE MAXIMUM AMPLITUDE OF h_n.

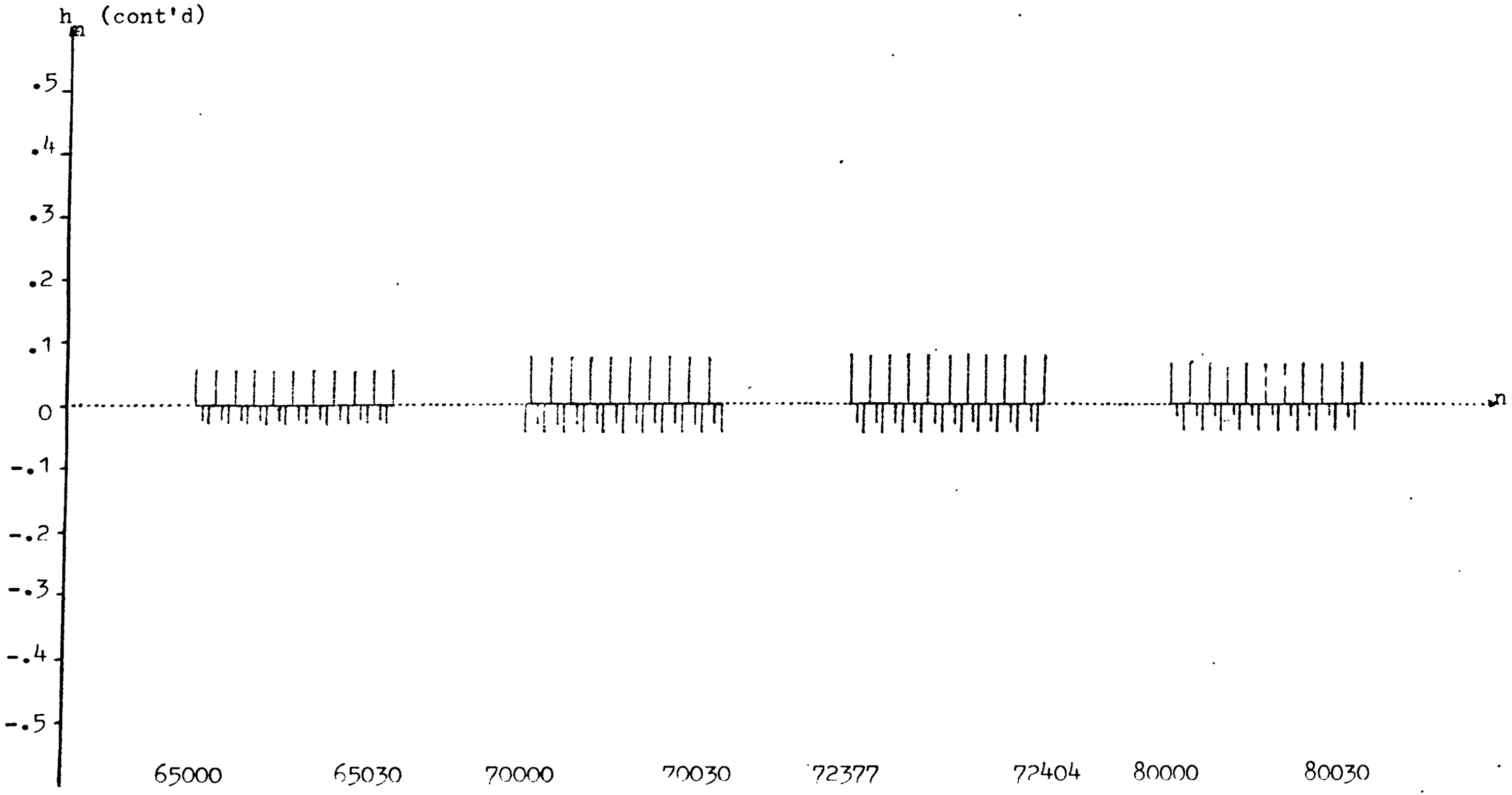




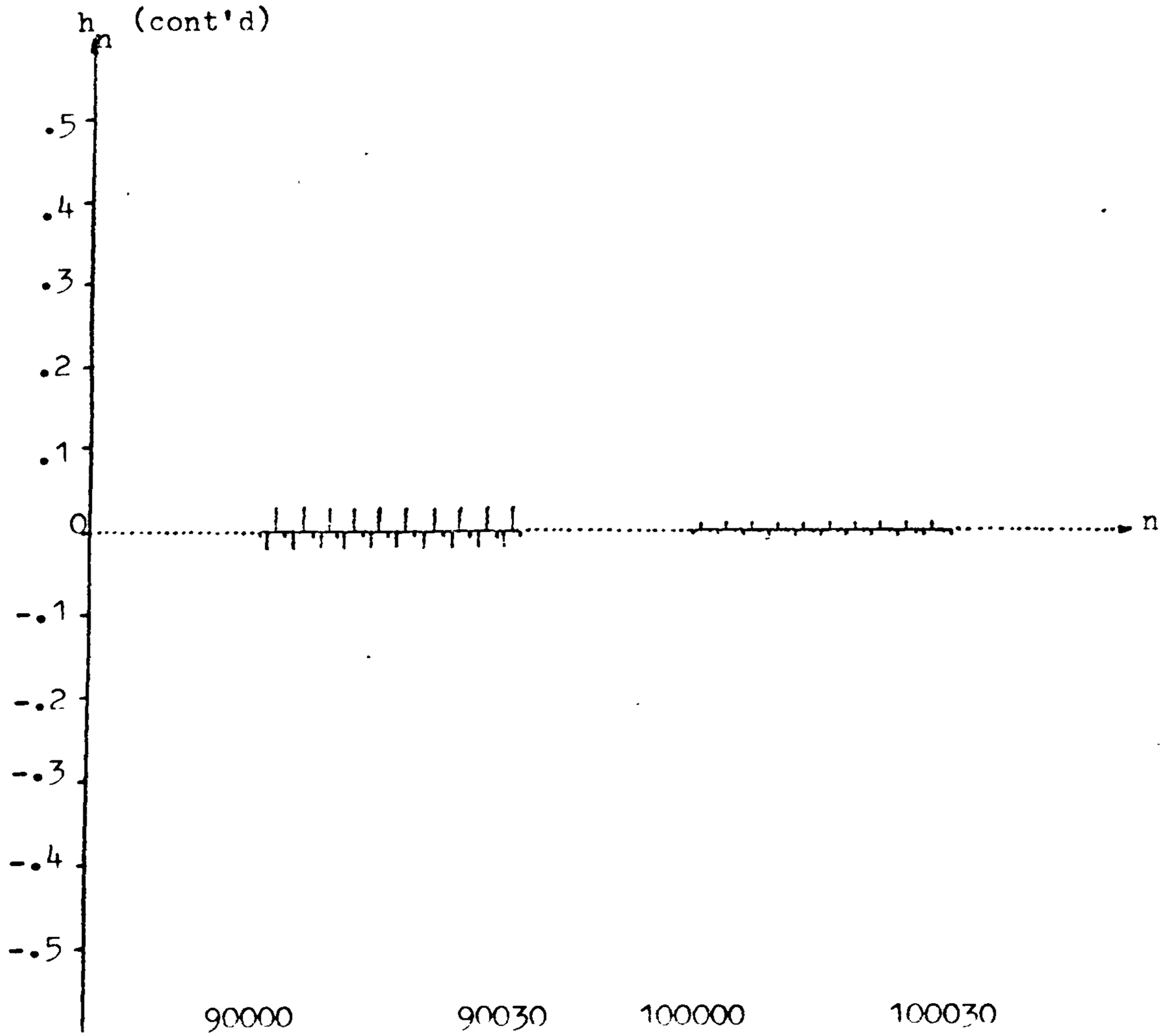
CORRELATION OF SIGNALS 27-28 FROM 31000 TO 54030 SHOWING THE DECAY OF h_n.

CONTINUATION OF FIGURE 8.2.27 SHOWING THE BUILD-UP OF THE RINGING OF h_n.





CONTINUATION OF FIGURE 8.2.27 SHOWING THE MAXIMUM AMPLITUDE OF THE RINGING OF h_n.



CONTINUATION OF FIGURE 8.2.27 SHOWING THE DECAY OF THE RINGING OF h_n.

shows the simulated impulse response of the above eighth-order bandpass filter centred at 1MHz with optimized transition sample W_n quantized to a value of 0.453125 and filter variables represented with infinite precision. Thus, the signals within the filter have a much larger dynamic range. Due to the fact that the bandwidth of this bandpass filter is very narrow, the rise time of the impulse response is extremely long. For instance, as shown in figure 8.2.27, the impulse response h_n of the above bandpass filter rises to its maximum value at approximately the 30000th sample. Such a long build up is well expected from the inverse proportionality of the rise time of a system to its bandwidth. That is, the narrower the bandwidth, the longer is the rise time. Such an observation above is further substantiated by figures 8.2.19 to 8.2.21 where it has been noticed that the impulse responses shown there are of much shorter durations.

It is further noticed from figure 8.2.27 that the impulse response h_n of this 1MHz bandpass filter also exhibits a ringing effect at approximately the 59000th sample which builds up to its maximum value at approximately the 70000th sample. This has been explained before to be due to the steep transition of the skirt of the frequency response of the 1MHz bandpass filter. Moreover, an anti-symmetry about this impulse response h_n is also observed. This, as explained before, indicates that, except for the ringing effect, the above 1MHz bandpass filter exhibits a piecewise-linear phase response. Of course, exact phase linearity is not achieved due to the presence of the above ringing effect.

8.2.6 EXPERIMENTAL RESULTS PERTINENT TO THE APPLICATION OF THE PROPOSED "SELECTIVE ROUNDING SCHEME":

The proposed "Selective Rounding Scheme" was discussed in section 6.8 in chapter six and its hardware implementation was discussed in section 7.14 in chapter seven. In this section, we shall present some results relating to the above scheme. Consider a second-order elemental filter centred at 1000100Hz, with the following transfer function:

$$H_Q(z) = \frac{0.900024414 + 0.4750976562z^{-1}}{1 + 0.950256348z^{-1} + 0.902221679 8z^{-2}} \qquad (8.2.11)$$

Figure 8.2.28 shows the experimental impulse response h_n of $H_Q(z)$ in equation (8.2.11) above. The input impulse had a magnitude of 0.96875 while the CBFCS memory used is shown in table C17 in appendix C. The impulse response dies down to a limit cycle of period three. Figure 8.2.29 shows the experimental impulse response of $H_Q(z)$ under the same input conditions but with the ordinary two's complement up-rounding unit RU of the demonstration processor being replaced by the "Selective Rounding Unit" (SRU) as described in section 7.14. From figure 8.2.29, it is noticed that the limit cycle behaviour of $H_Q(z)$ is completely eliminated under the use of the proposed "Selective Rounding Scheme".

In the application of the above "Selective Rounding Scheme", in addition to injecting random rounding noise, certain output states of the demonstration processor are selectively forbidden, and hence the name of the proposed scheme. As described in section 7.14, the "Selective Rounding Unit" (SRU) can add or subtract random noise, according to a selective rounding function (SRF), to the rounding operations of the demonstration processor. The selective rounding function (SRF) for the above second-order elemental filter is given by equation (7.14.5) in section 7.14. Owing to the use of this selective rounding function (SRF), a random error sequence whose elements are members of the random error set:

$$\{(2^{-7}+2^{-8}),\ -2^{-8},\ 2^{-8}\} \qquad (8.2.12)$$

results. This is then the random roundoff error set of the proposed "Selective Rounding Scheme" for the above second-order elemental filter which exhibits a limit cycle behaviour when ordinary two's complement up-rounding is used. To complete the above "Selective Rounding Scheme", the output state (1.1111111) of the demonstration processor was also forbidden for the above second-order elemental filter. In so doing, the output of the demonstration processor was repeatedly set to (0.0000000) whenever the output state (1.1111111) appeared. This then successfully broke up the small-amplitude noise like output due to the application of the above random rounding process in such a way that the experimental impulse response shown in figure 8.2.29 actually died down to zero value.

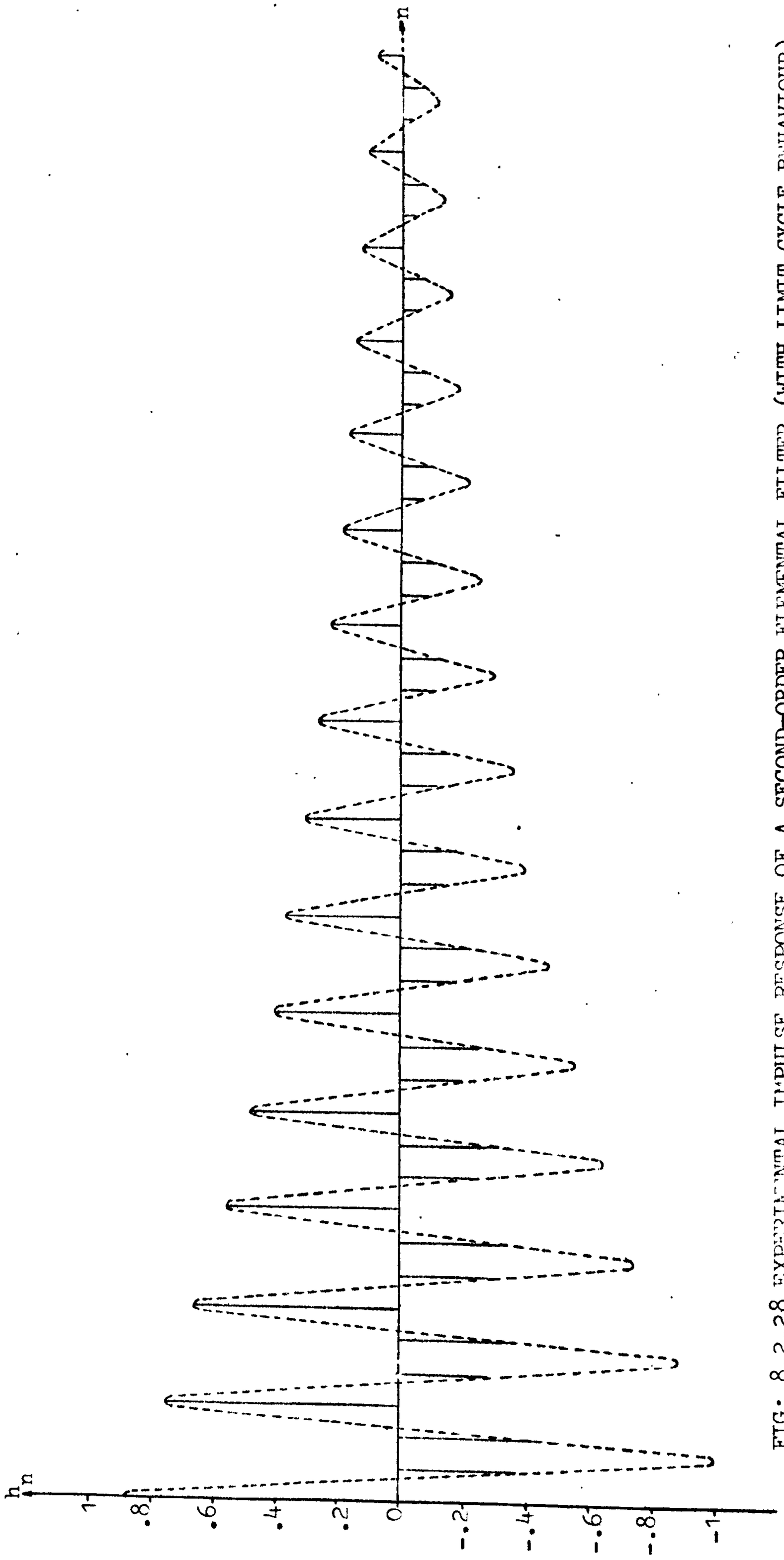
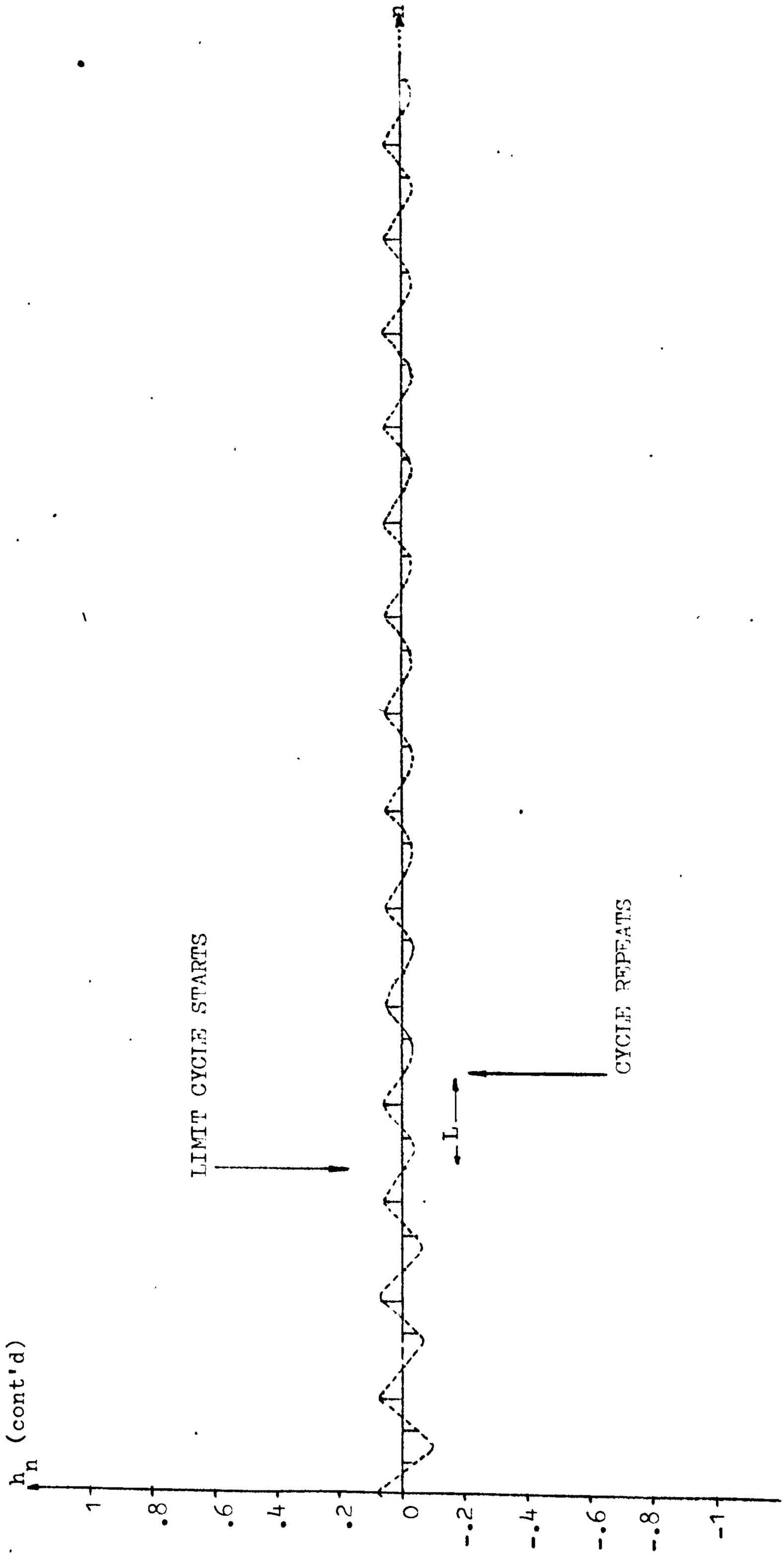
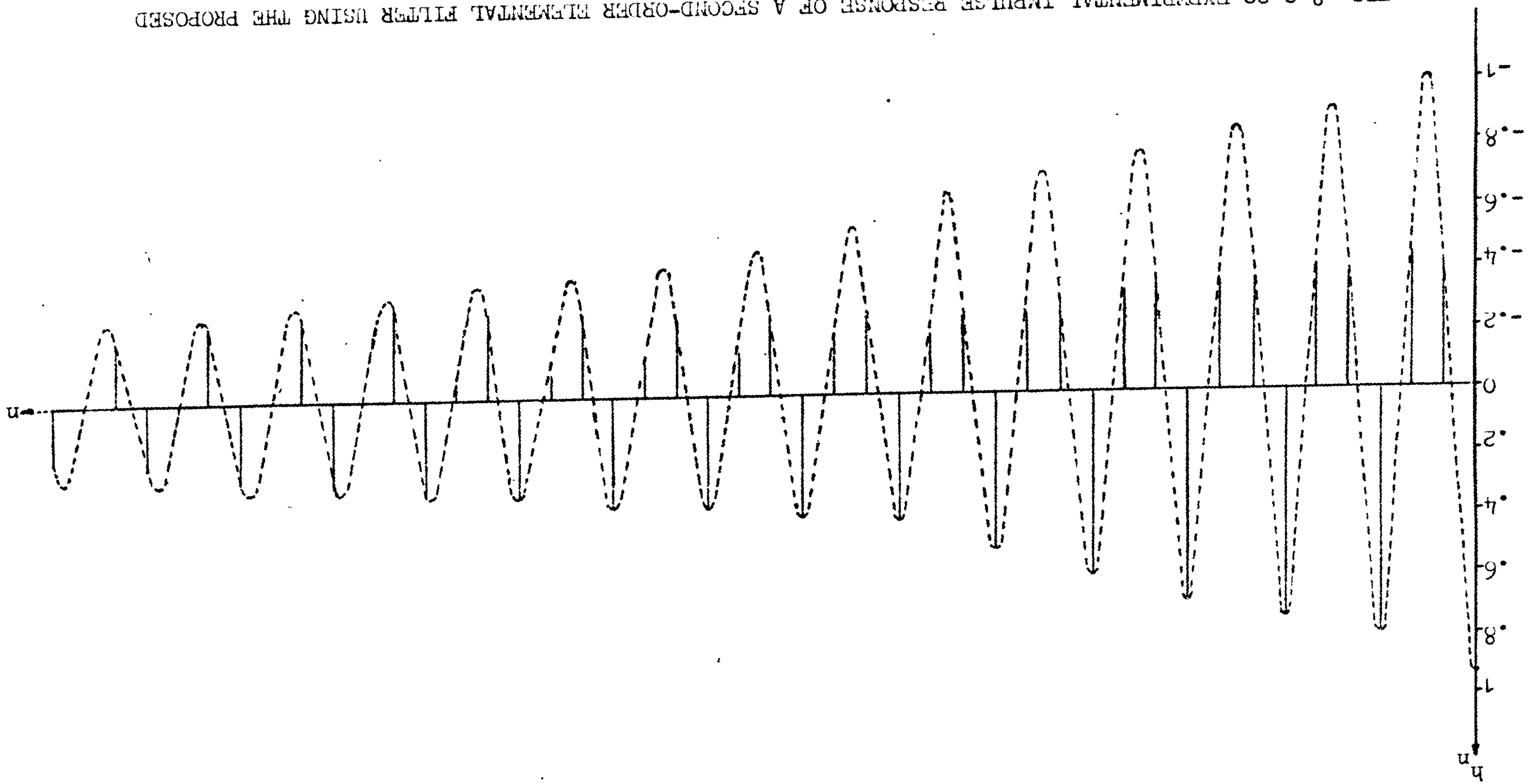


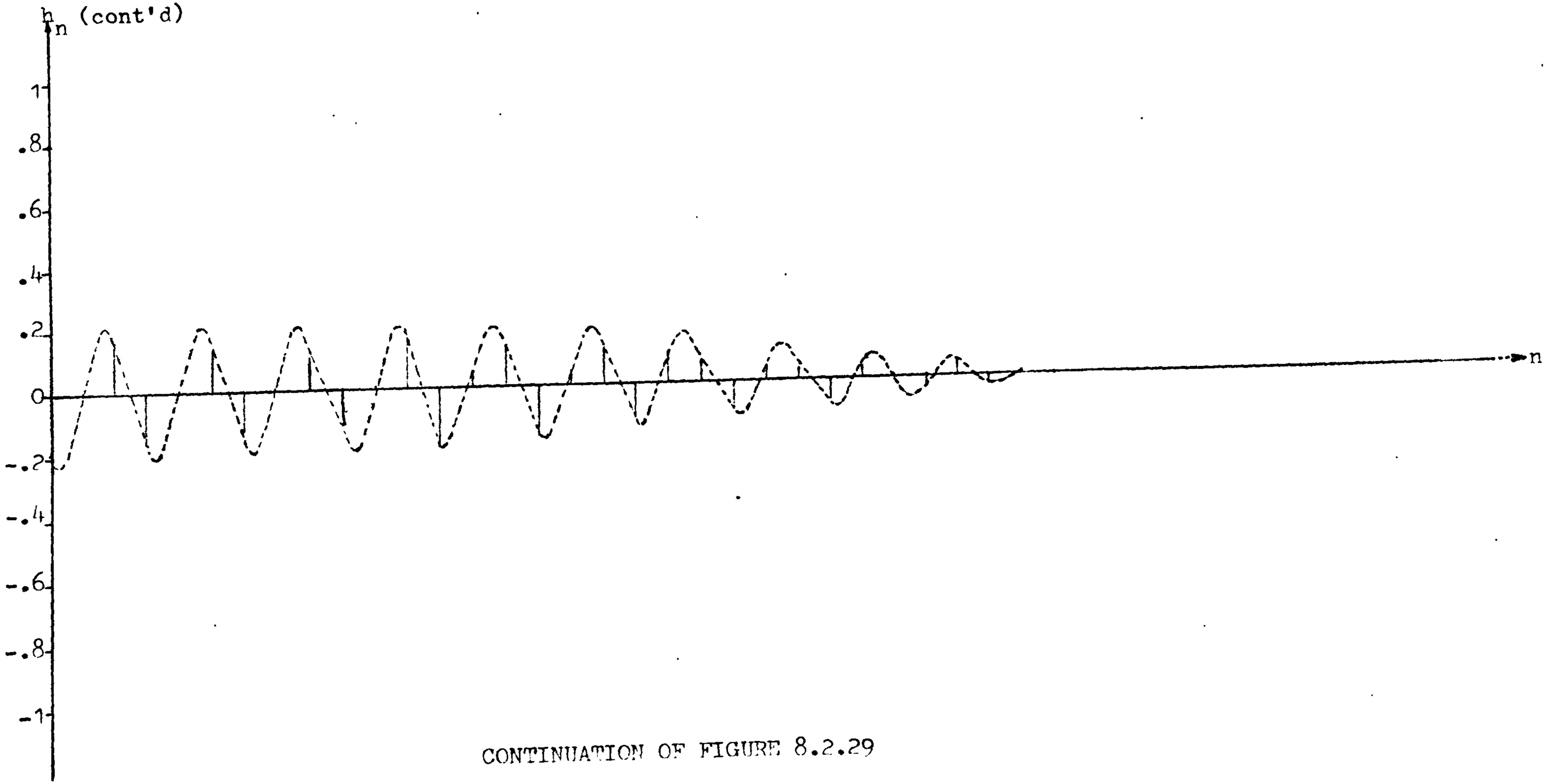
FIG: 8.2.28 EXPERIMENTAL IMPULSE RESPONSE OF A SECOND-ORDER ELEMENTAL FILTER (WITH LIMIT CYCLE BEHAVIOUR).



CONTINUATION OF FIGURE 8.2.28.

FIG: 8.2.29 EXPERIMENTAL IMPULSE RESPONSE OF A SECOND-ORDER ELEMENTAL FILTER USING THE PROPOSED "SELECTIVE ROUNDING SCHEME".





CONTINUATION OF FIGURE 8.2.29

However, as mentioned in section 6.8, the use of the above "Selective Rounding Scheme" would introduce more quantization error in the filtering process, as can be observed in the latter part of the impulse response h_n shown in figure 8.2.29 which is somewhat prolonged due to the non-linearity introduced by the set of random roundoff errors. In general, such a scheme should be avoided in systems where idle channel noise is not of significance. On the other hand, in systems where idle channel noise has to be kept to a minimum, the above scheme would be applicable.

So far, we have discussed the choice between the realization configurations shown in figure 8.1.2. Experimental impulse responses and simulated impulse responses of several filters have all been presented and discussed. In addition, the limit cycle behaviours of these filters have been analyzed using the generalized noise model discussed in chapter six. Furthermore, the use of the proposed "Selective Rounding Scheme" to eliminate limit cycle behaviours of filters implemented by the proposed CBFCS approach has also been discussed, while experimental results have further verified the proposed scheme. In the next section, we shall present the frequency responses of some filters (both experimental and simulated) designed by the proposed design technique discussed in chapter three.

8.3 FREQUENCY DOMAIN RESPONSES:

In this section, the experimental frequency response of bandpass filter A, centred at 120Hz, whose experimental impulse response was shown in figure 8.2.5 in section 8.2, is first presented. Simulated frequency response of the same bandpass filter is then given for comparison and analysis purposes. An analysis of the finite word length effects on the above experimental frequency responses is then given, based on the proposed generalized quantization noise model discussed in chapter six. Finally, further simulation results are presented for the 1MHz design in section 8.2.4, since experimental results were not obtained due to the limited word length of the demonstration processor and the limitation in speed of its bipolar analogue-to-digital converter (A/D) as explained in section 7.15 in chapter seven. Additional comments and relevant conclusions are given wherever appropriate.

8.3.1 EXPERIMENTAL FREQUENCY RESPONSES OF A BANDPASS FILTER AND ITS CONSTITUENT ELEMENTAL FILTERS:

In this section, the experimental frequency responses and phase responses of the previously designed bandpass filter A and its constituent elemental filters are presented. Bandpass filter A was chosen because of the fact that configuration (b) shown in figure 8.1.2 was shown to be the best realization scheme as far as finite word length effects were concerned.

In obtaining the above mentioned experimental frequency responses, the input signal x_n and the quantized transfer function $H_Q(z)$ of bandpass filter A were scaled to prevent overflow due to the gains of the individual elemental filters. The scaling of $H_Q(z)$ of bandpass filter A, represented by equation (8.2.2), was done by scaling the transfer functions of the individual elemental filters. (Due to this scaling, the corresponding contents of the CBFCS memories shown in tables C1 to C4 were accordingly altered). Figures 8.3.1 to 8.3.5 show the experimental frequency and phase responses of the above bandpass filter A and its constituent elemental filters.

Figure 8.3.1 shows the frequency and phase responses of the second-order lower transition elemental filter in the lower transition band with the optimized transition sample W_n quantized to a value of 0.5 as discussed in the previous section (see equation (8.2.2) in section 8.2.1). Since the above elemental filter, $H_1^T(z)$, is basically a resonator, its frequency response exhibit a resonance at its resonance frequency 115Hz. This follows from the analysis shown in appendix B. Owing to the transition sample W_n=0.5, the peak gain of $H_1^T(z)$ is observed from figure 8.3.1 to be -6dB approximately. The nominal 3dB bandwidth of $H_1^T(z)$ is seen to be approximately 5Hz. The phase response of $H_1^T(z)$ is approximately linear within the passband of $H_1^T(z)$.

Figure 8.3.2 shows the frequency and phase responses of the second-order first passband elemental filter $H_1^P(z)$ in the passband. Since transition poles are seperated each from its nearest passband pole by 2.5Hz in the design, the resonance frequency of $H_1^P(z)$ is 117.5Hz. The nominal 3dB bandwidth of $H_1^P(z)$ is seen to be 5Hz

approximately. The phase response of $H_1^P(z)$ is approximately linear within the passband of $H_1^P(z)$. Figure 8.3.3 shows the frequency and phase responses of the second second-order passband elemental filter $H_2^P(z)$ in the passband. The resonance frequency of $H_2^P(z)$ is 122.5Hz, and, in fact, it is noticed that the frequency responses of the two passband elemental filters $H_1^P(z)$ and $H_2^P(z)$ overlap on each other at their 3dB points. As in the case of the first passband elemental filter $H_1^P(z)$, the nominal 3dB bandwidth of $H_2^P(z)$ is seen to be 5Hz approximately. The phase response of $H_2^P(z)$ is approximately linear within the passband of $H_2^P(z)$.

Figure 8.3.4 shows the frequency and phase responses of the second-order upper transition elemental filter in the upper transition band with the optimized transition sample W_n quantized to a value of 0.5 as discussed in the previous section (see equation (8.2.2) in section 8.2.1). The resonance frequency of the above elemental filter $H_u^T(z)$ is at 125Hz. Owing to the transition sample $W_n = 0.5$, the peak gain of $H_u^T(z)$ is observed from 8.3.4 to be -6dB approximately. The nominal 3dB bandwidth of $H_u^T(z)$ is seen to be approximately 5Hz. The phase response of $H_u^T(z)$ is approximately linear within the passband of $H_u^T(z)$.

As previously described and discussed, the above four elemental filters are "phase-modified" (see section 3.2.2) in such a way that their responses add up to that as shown in figure 8.3.5. As shown in figure 8.3.5, the frequency response of the bandpass filter A, centred at 120Hz, with a word length of eight bits including sign for the filter variables, has a 3dB bandwidth of approximately 10Hz. In the next section, it will be shown that the frequency responses shown in this section in figures 8.3.1 to 8.3.5 vary from the ideal responses with infinite accuracy. The discrepancies are due to the effects of finite word lengths. In section 8.3.3, the errors due to the use of finite word lengths will be analyzed with the proposed generalized quantization model discussed in section 6.6 in chapter six. Finally, it is noticed from figure 8.3.5 that the phase response of bandpass filter A resembles a piece-wise linear characteristic, except at the bandedges where non-linearities are obvious and mainly due to the optimization of the transition poles. In the next section,

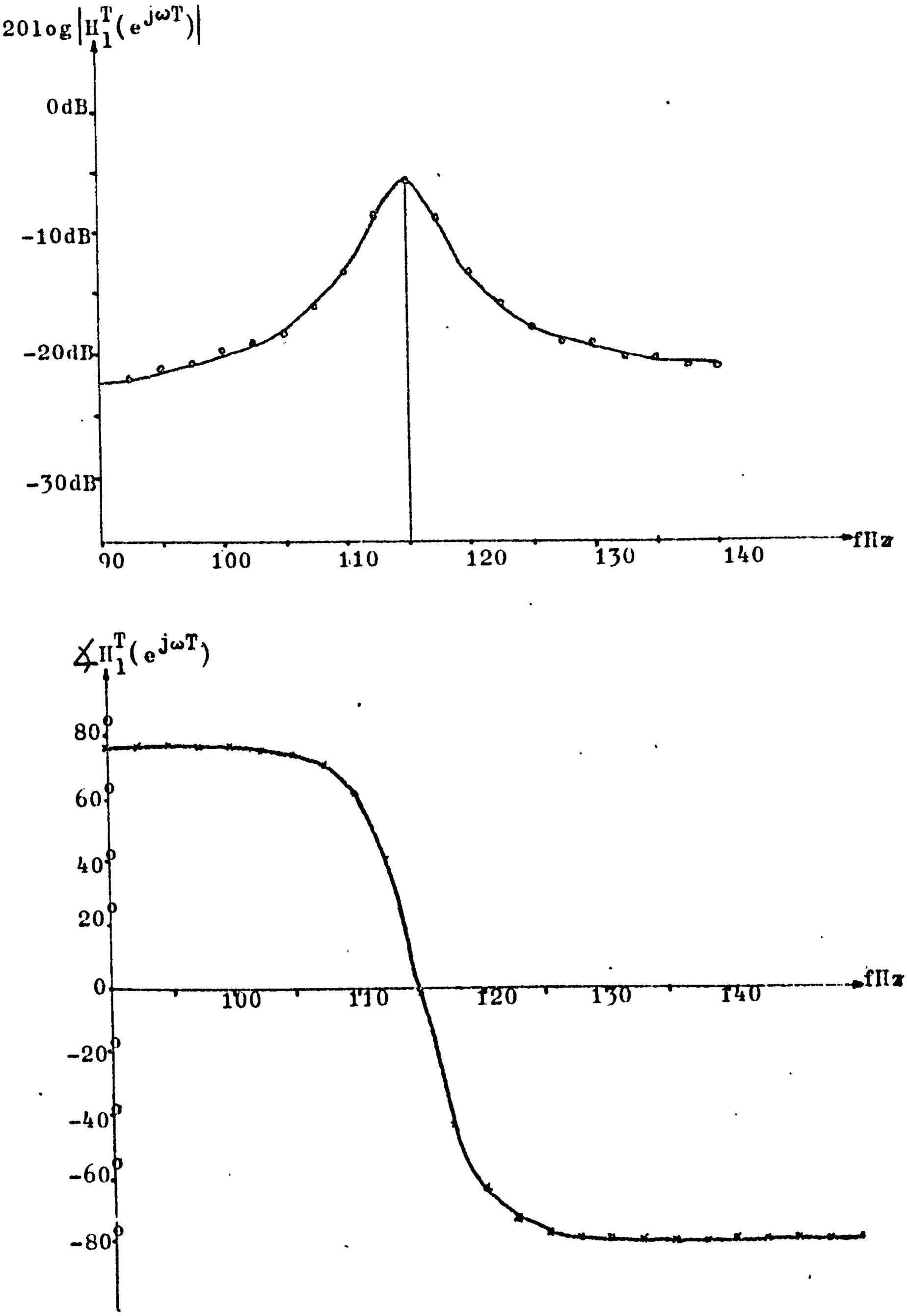
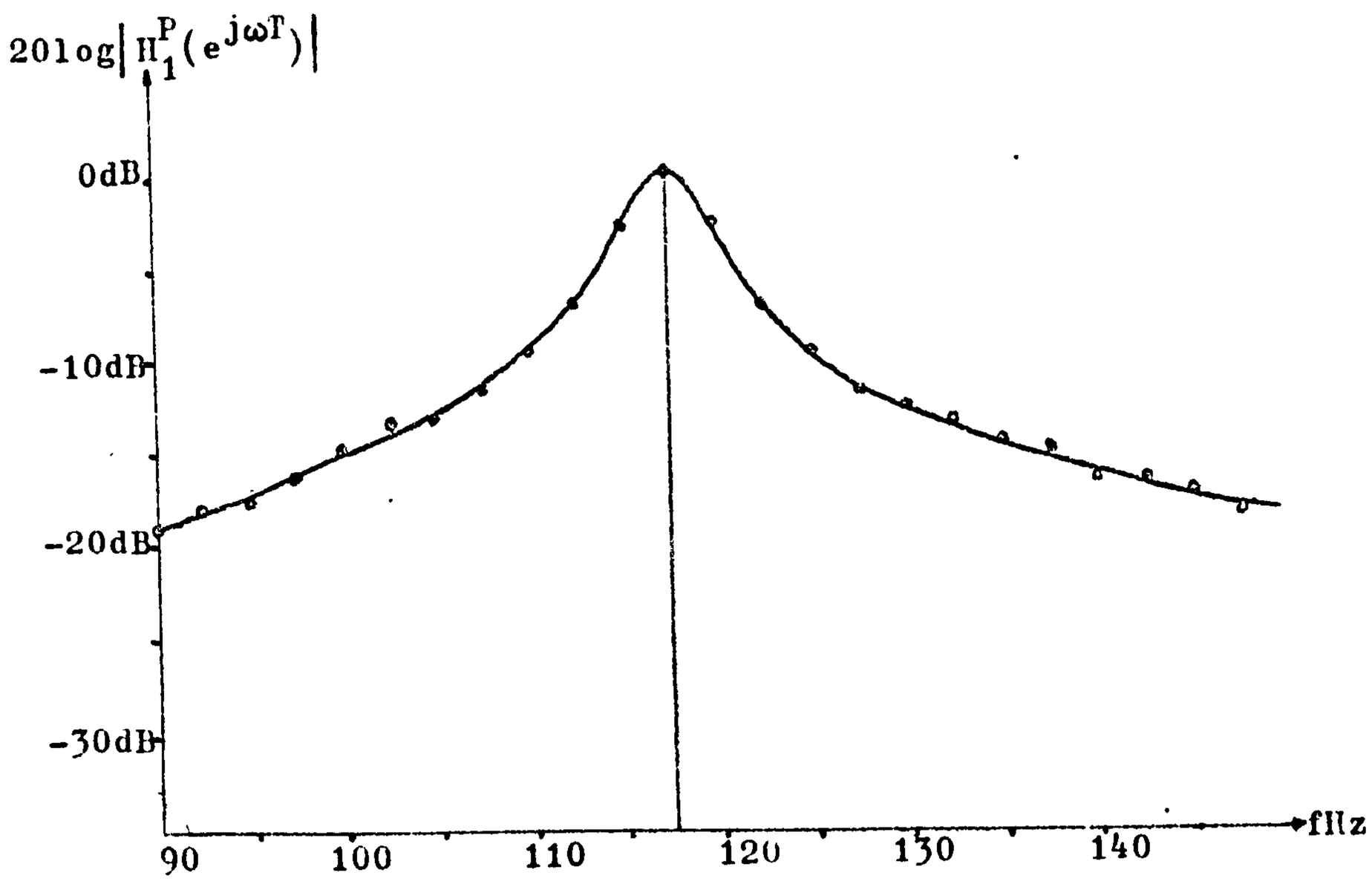


FIG: 8.3.1 FREQUENCY AND PHASE RESPONSES OF LOWER TRANSITION ELEMLNTAL FILTER $H_1^T(z)$, $W_n = 0.5$ (WORDLENGTH: 8 BITS INCLUDING SIGN).



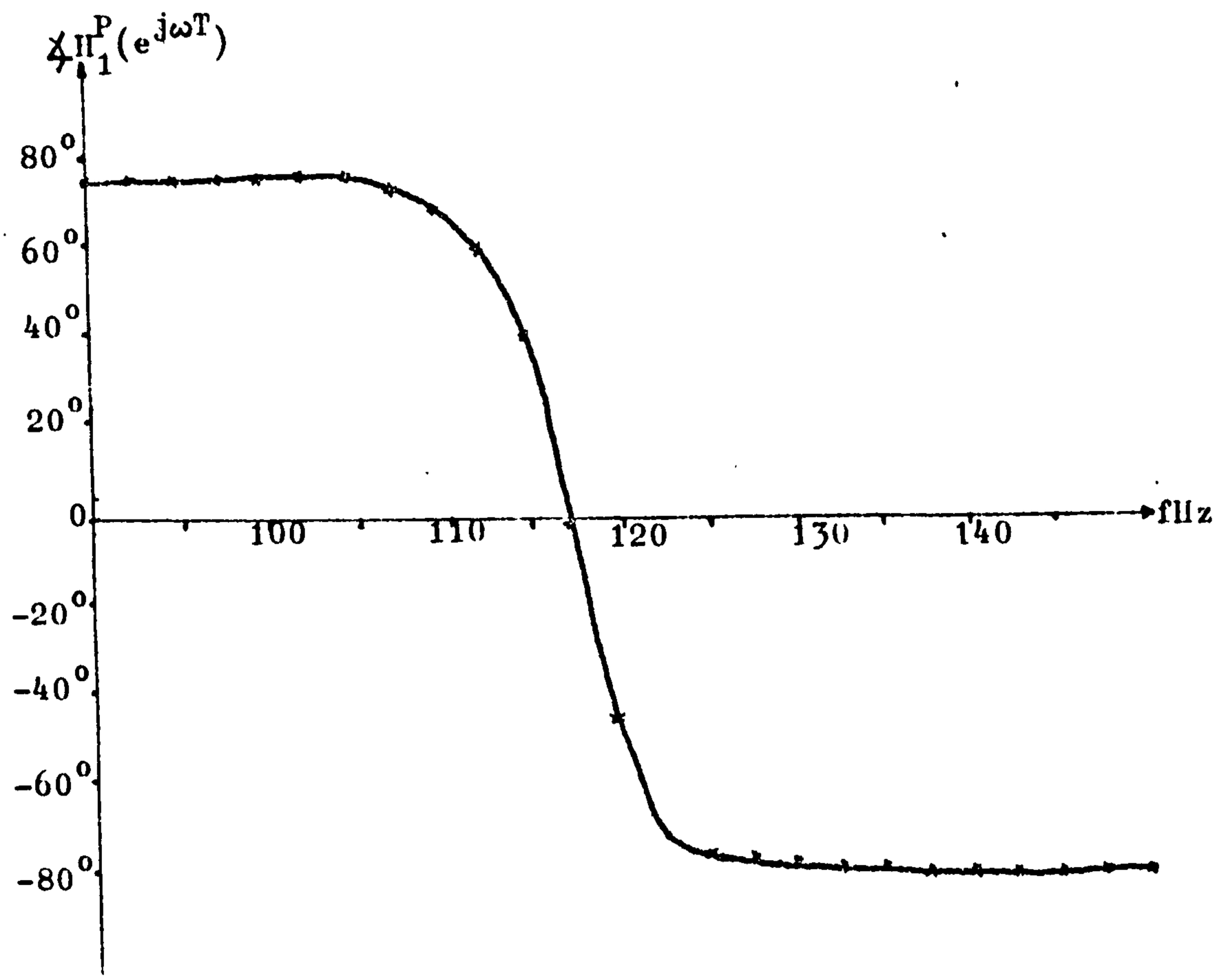


FIG: 8.3.2 FREQUENCY AND PHASE RESPONSES OF THE FIRST PASSBAND ELEMENTAL FILTER $H_1^P(z)$ (WORDLENGTH: 8 BITS INCLUDING SIGN).

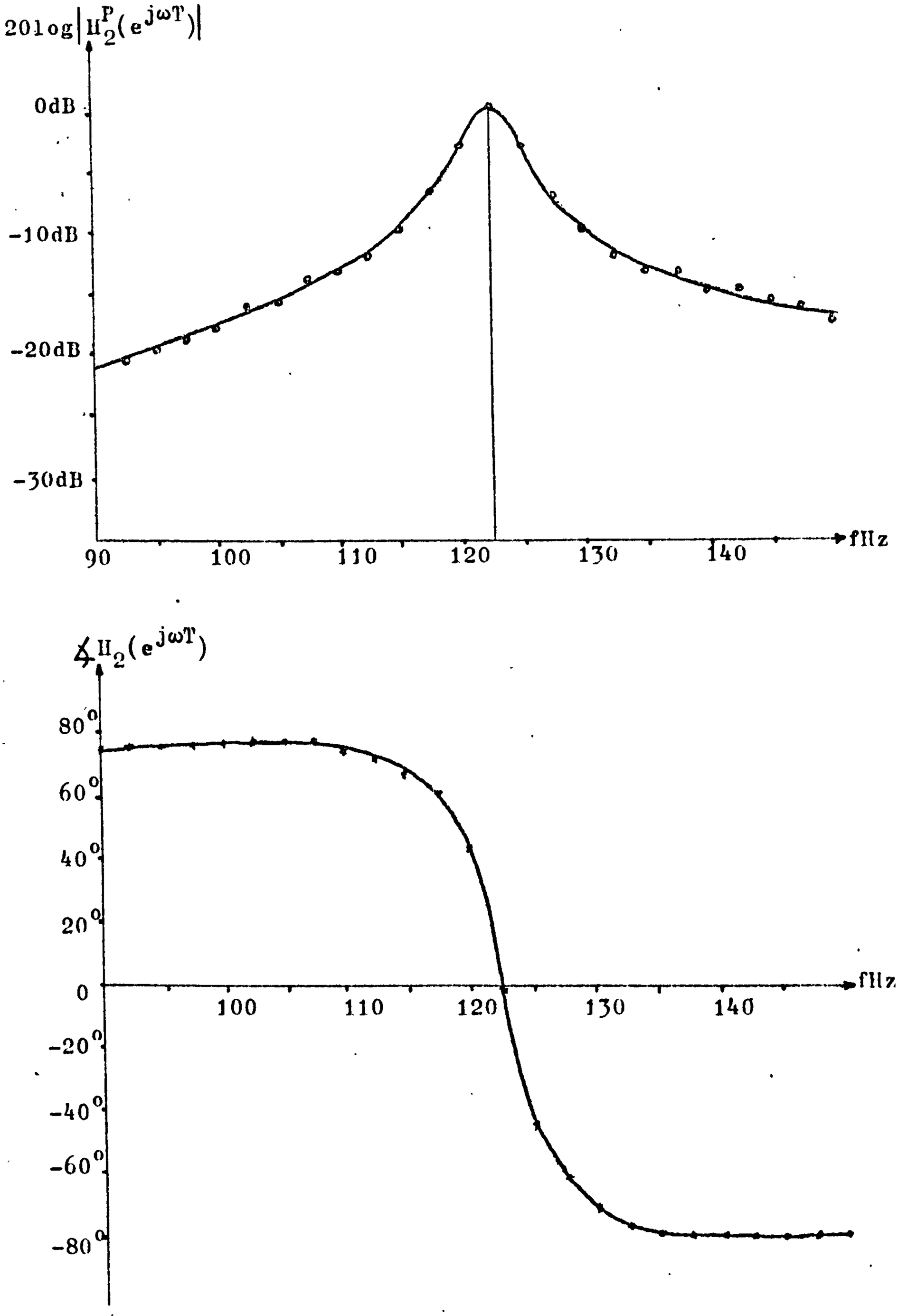
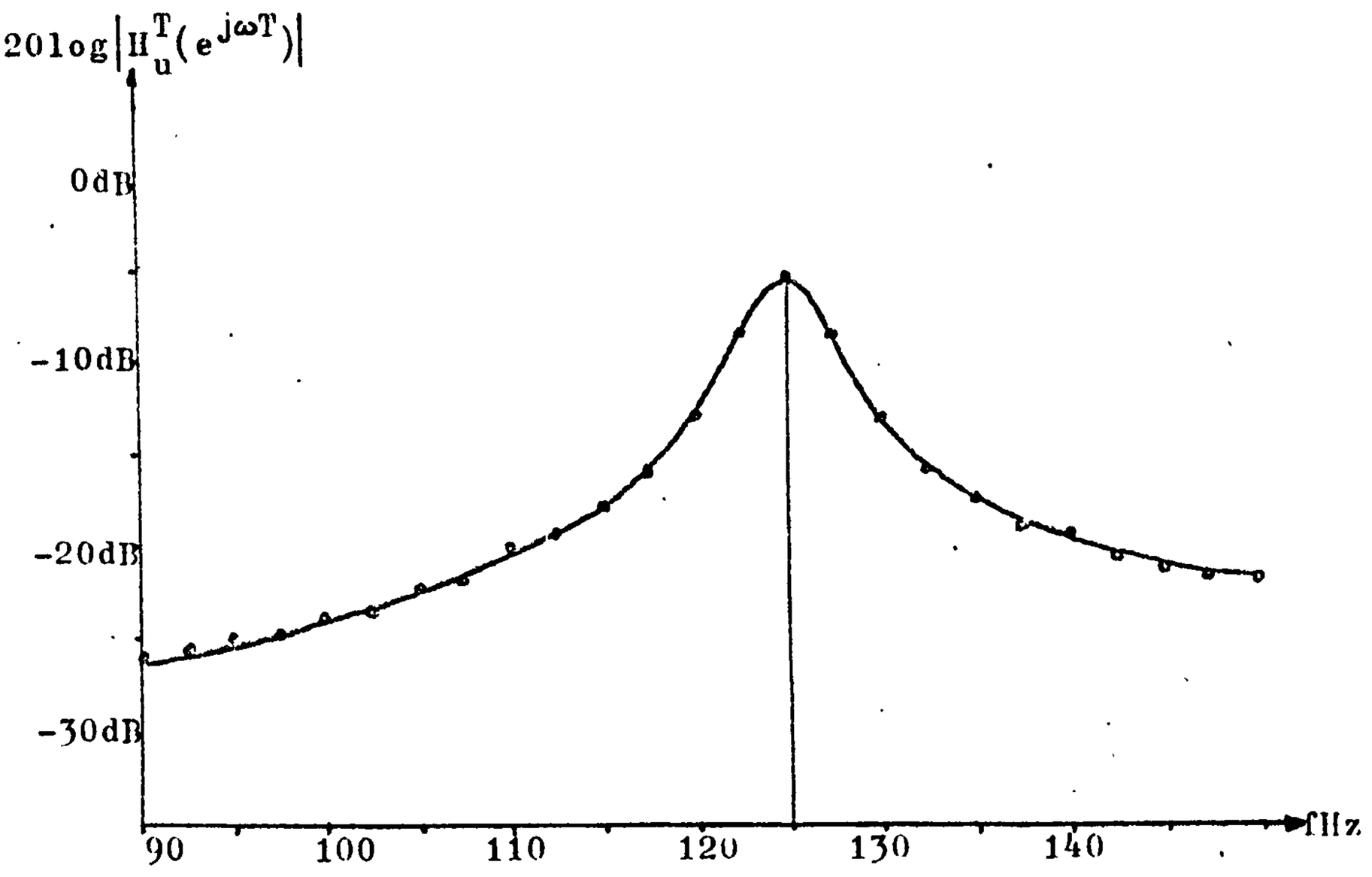


FIG: 8.3.3 FREQUENCY AND PHASE RESPONSES OF THE SECOND PASSBAND ELEMENTAL FILTER $H_2^P(z)$ (WORDLENGTH: 8 BITS INCLUDING SIGN).



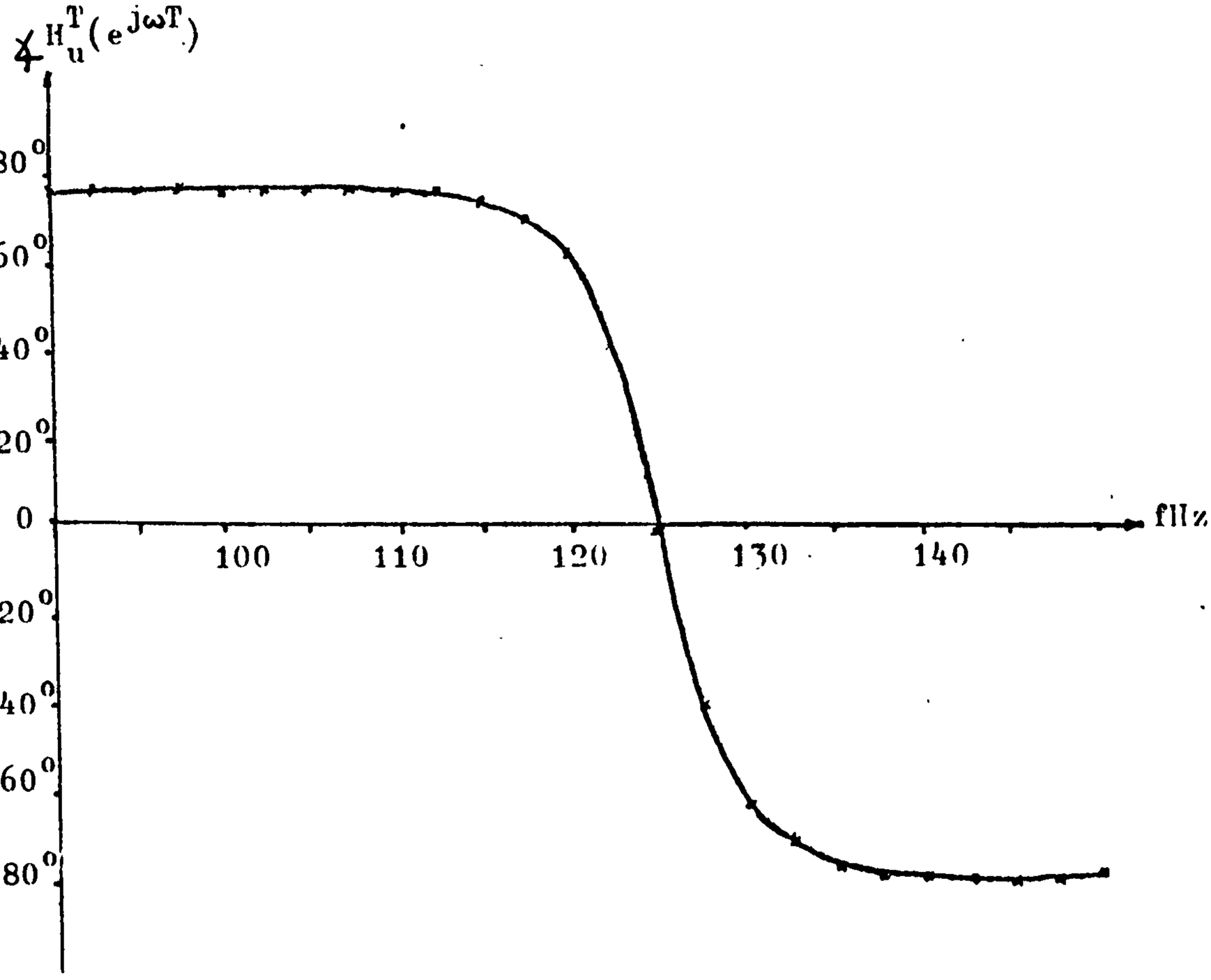
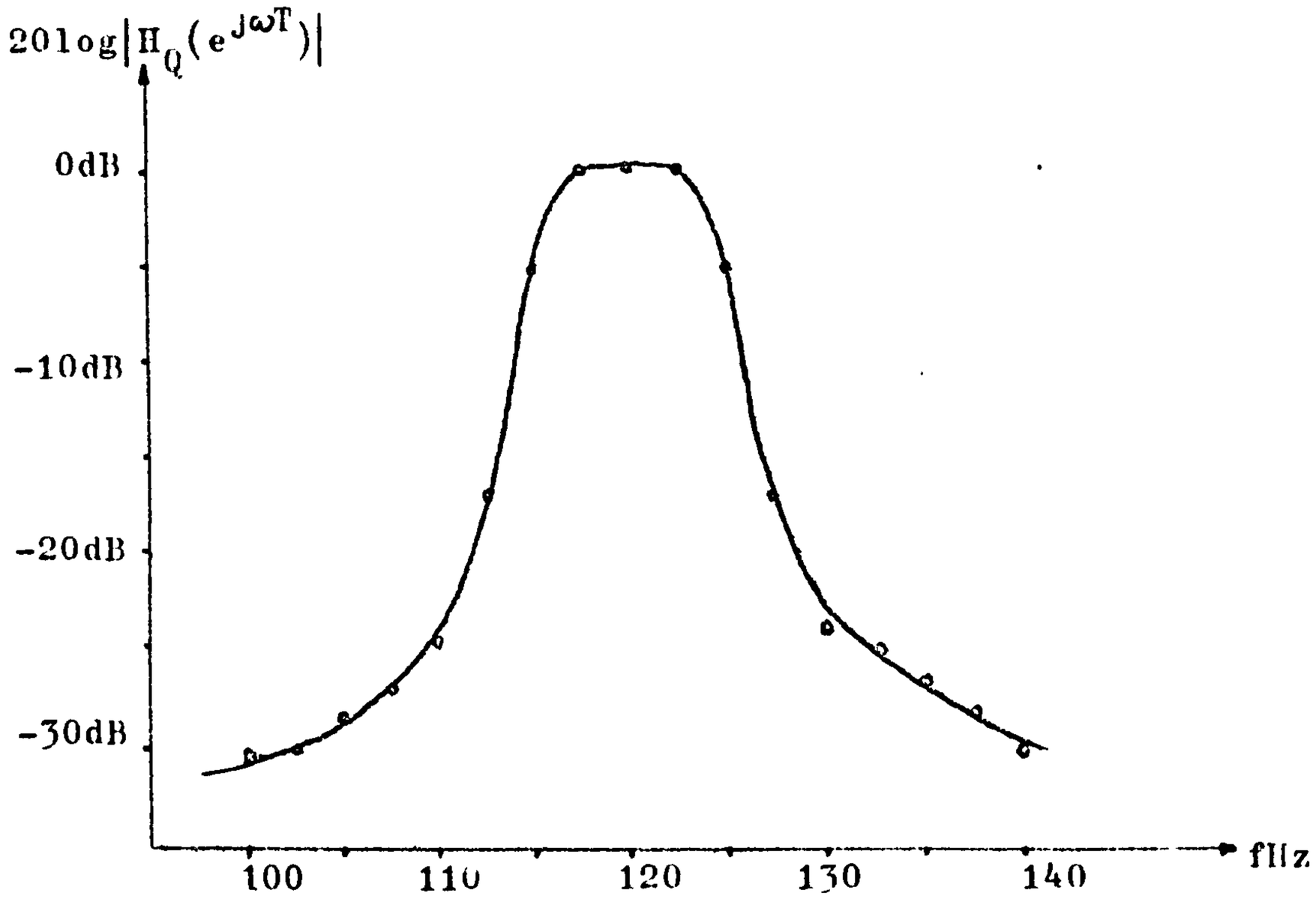


FIG: 8.3.4 FREQUENCY AND PHASE RESPONSES OF UPPER TRANSITION ELEMENTAL FILTER $H_u^T(z)$, $W_n = 0.5$ (WORDLENGTH: 8 BITS INCLUDING SIGN).



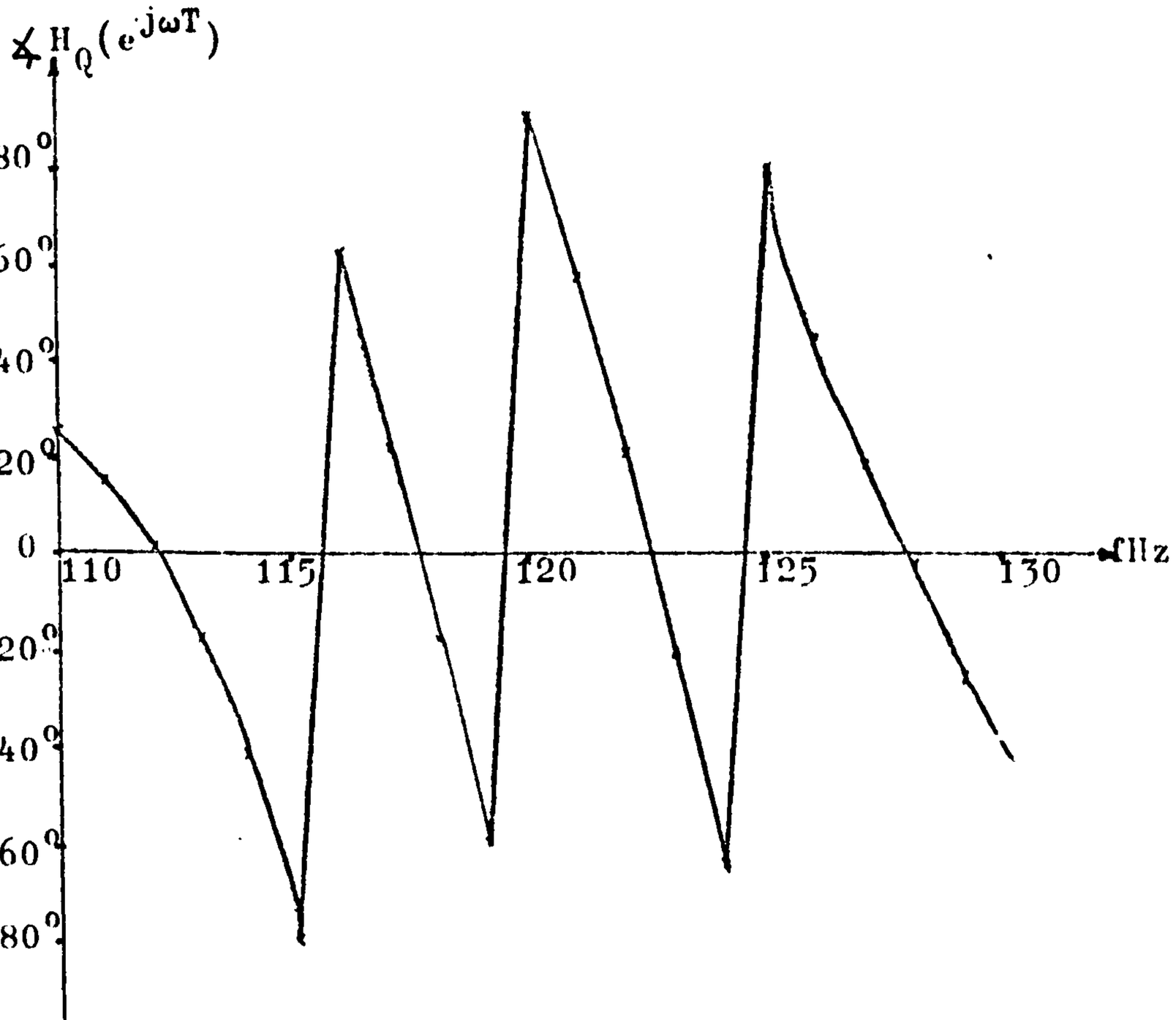


FIG: 8.3.5 FREQUENCY AND PHASE RESPONSES OF A BANDPASS FILTER CENTRED AT 120Hz (WORDLENGTH: 8 BITS INCLUDING SIGN).

it will be seen that the ideal phase response of bandpass filter A, using infinite word lengths for its filter variables, exhibits a high degree of piecewise linearity as expected from an examination of its impulse response, using infinite word lengths for its filter variables, shown in figure 8.2.19 and discussed in section 8.2.2. The distortion of the phase response as shown in figure 8.3.5 above is also expected from an examination of the impulse response of bandpass filter A, using eight bits including sign for its filter variables, shown in figure 8.2.5.

8.3.2 SIMULATED FREQUENCY RESPONSES PERTINENT TO SECTION 8.3.1:

In this section, we present the simulated frequency responses of bandpass filter A and its constituent elemental filters based on the simulation model shown in figure 8.1.4. Figures 8.3.6 to 8.3.10 shown the ideal frequency responses of bandpass filter A and its constituent elemental filters simulated using infinite word lengths for the filter variables. In the following, comparisons are made between the above mentioned ideal frequency responses and those shown in the last section (with a wordlength of eight bits including sign) to illustrate the effects of finite word lengths.

Figure 8.3.6 shows the ideal frequency response of the lower transition elemental filter $H_1^T(z)$ of bandpass filter A. A comparison between figures 8.3.1 and 8.3.6 shows that the frequency response of $H_1^T(z)$ with a word length of eight bits including sign differ from that with an infinite word length in their respective out-of-band rejections achieved. For instance, at 90Hz, the out-of-band rejection of $H_1^T(z)$ as shown in figure 8.3.6 is approximately -26.6dB while that shown in figure 8.3.1 is only approximately -22.5dB. This distortion in the frequency response with a wordlength of eight bits including sign, is due to the effects of finite word lengths discussed in chapter six. This will be further discussed in the analysis of finite word length effects on the frequency response of bandpass filter A, with a word length of eight bits including sign, in the next section.

Figure 8.3.7 shows the ideal frequency response of the first passband elemental filter $H_1^P(z)$ of bandpass filter A. A comparison between figures 8.3.2 and 8.3.7 shows a similar distortion in frequency response in terms of achievable out-of-band rejection. For instance,

at 90Hz, the out-of-band rejection of the ideal frequency response of $H_1^P(z)$ is approximately -21.5dB while the achievable out-of-band rejection in that shown in figure 8.3.2 is only approximately -19dB. Again, this is due to finite word length effects, and will be further discussed in the analysis given in the next section.

Figure 8.3.8 shows the ideal frequency response of the second passband elemental filter $H_2^P(z)$ of bandpass filter A. A comparison between figure 8.3.3 and figure 8.3.8 shows the discrepancies in achievable out-of-band rejections due to finite word length effects. For instance, at 90Hz, the out-of-band rejection in figure 8.3.8 is approximately -23dB while that shown in figure 8.3.3 is only approximately -21.5dB. This will be further discussed in the next section.

Figure 8.3.9 shows the ideal frequency response of the upper transition elemental filter $H_u^T(z)$ of bandpass filter A. A comparison between figures 8.3.4 and 8.3.9 shows the discrepancies in achievable out-of-band rejections due to finite word length effects. For instance, at 90Hz, the out-of-band rejection shown in figure 8.3.9 is around -30dB while that shown in figure 8.3.4 is around -26dB only. This will be discussed in the next section.

Figure 8.3.10 shows the ideal frequency response of the above bandpass filter A. A comparison between figures 8.3.5 and 8.3.10 shows a more profound discrepancy in the two frequency responses shown, in terms of out-of-band rejections achievable. For instance, at 100Hz, the out-of-band rejection shown in figure 8.3.10 is around -52dB, however, that shown in figure 8.3.5 is only approximately -31dB only. Hence, a larger distortion results in the eight-order bandpass filter A, with a word length of eight bits including sign, than its constituent second-order elemental filters. This is expected, as discussed in chapter six, since the output error due to finite word length effects is larger for a higher-order filter than a lower-order filter. This will be further illustrated in the next section. Figure 8.3.11 shows the approximately piecewise linear phase response of the desired $H_Q(z)$ with infinite precision for its filter variables. As discussed in section 8.2.2, except for the ringing, the impulse response h_n of bandpass filter A with infinite word lengths for its filter variables shown in figure 8.2.19 exhibits

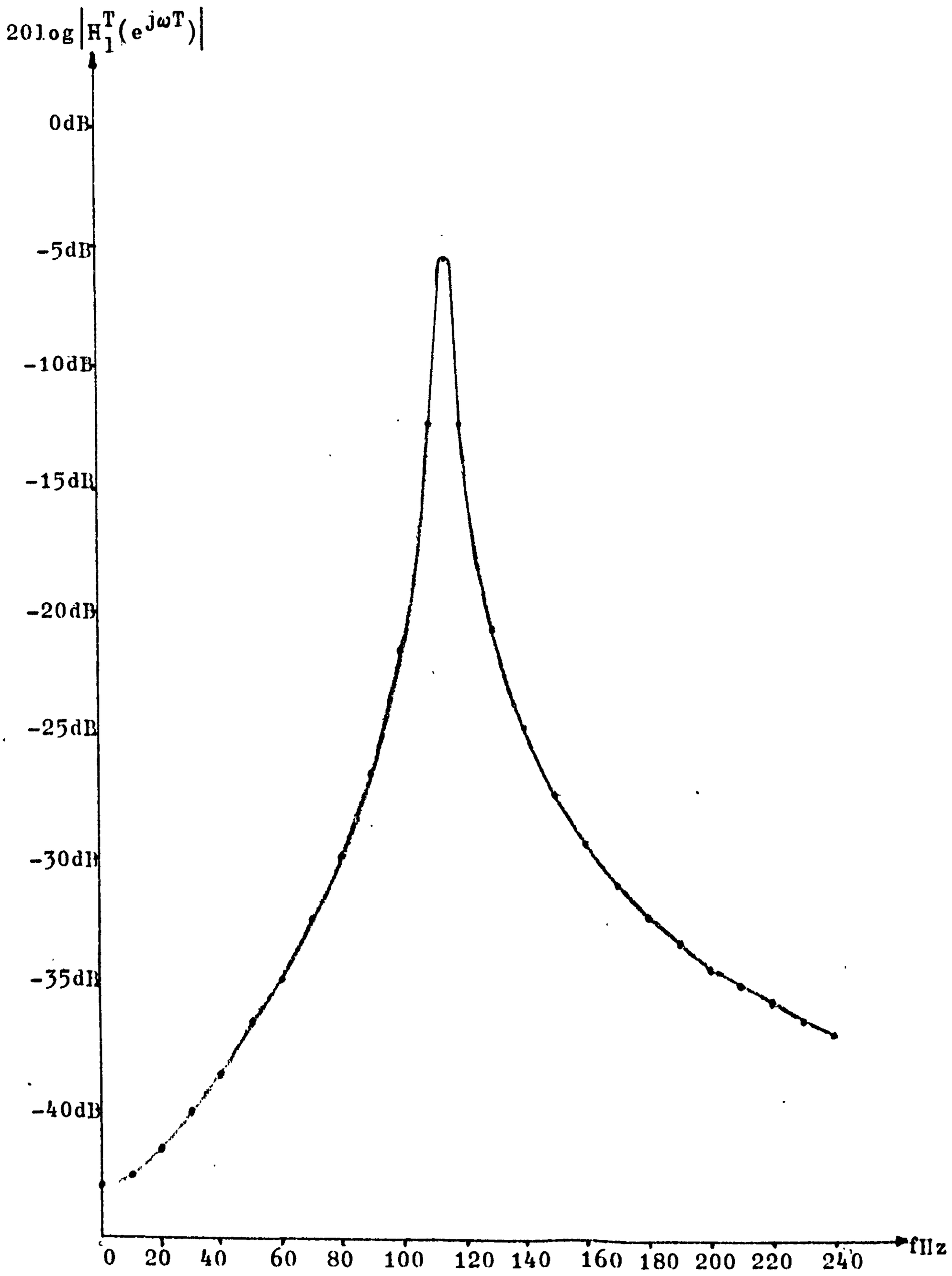


FIG: 8.3.6 FREQUENCY RESPONSE OF LOWER TRANSITION ELEMENTAL FILTER $H_1^T(z)$, $W_n = 0.5$ (WORDLENGTH: INFINITE).

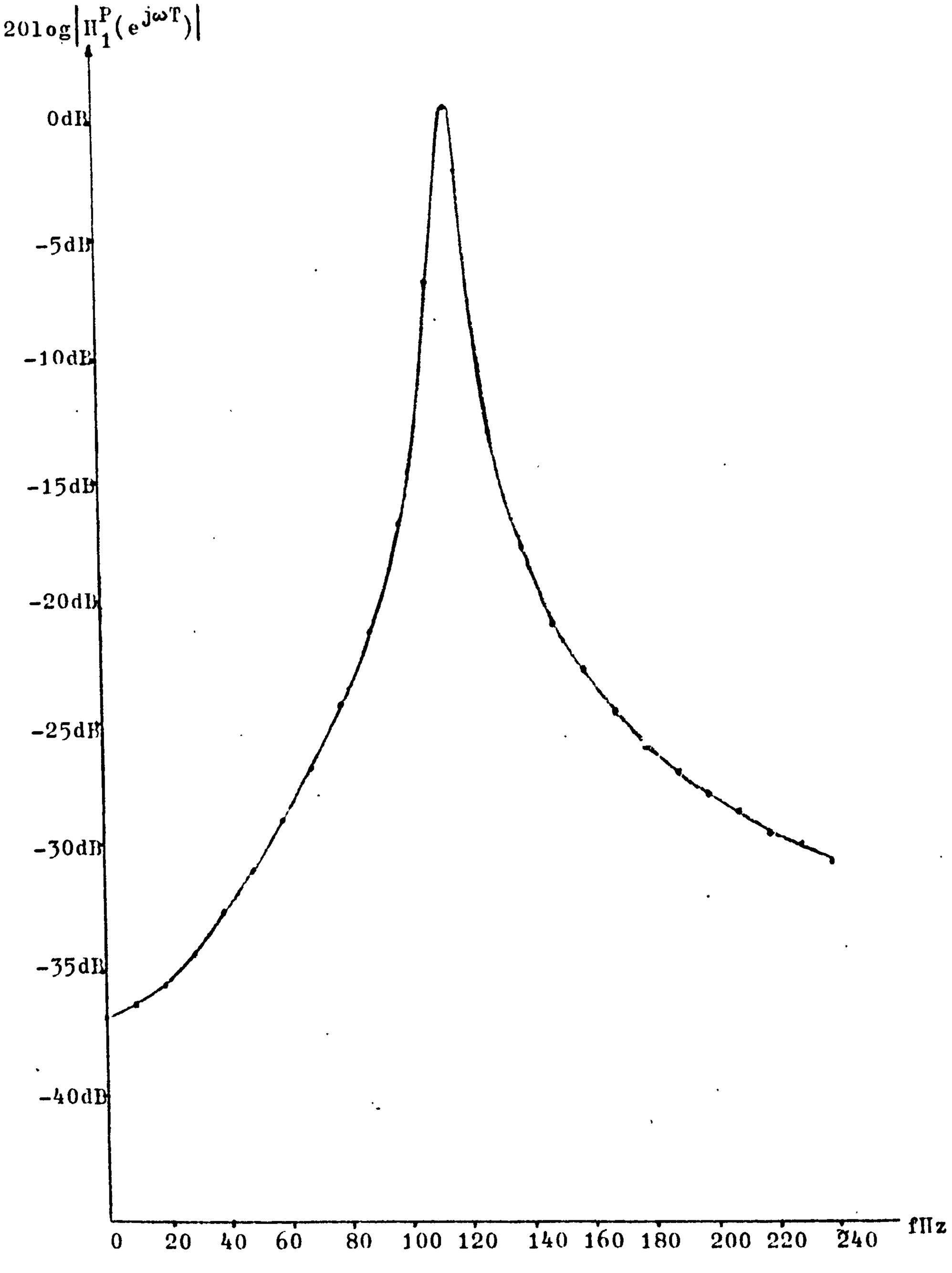


FIG: 8.3.7 FREQUENCY RESPONSE OF THE **FIRST** PASSBAND ELEMENTAL FILTER $H_1^P(z)$. (WORDLENGTH: INFINITE)

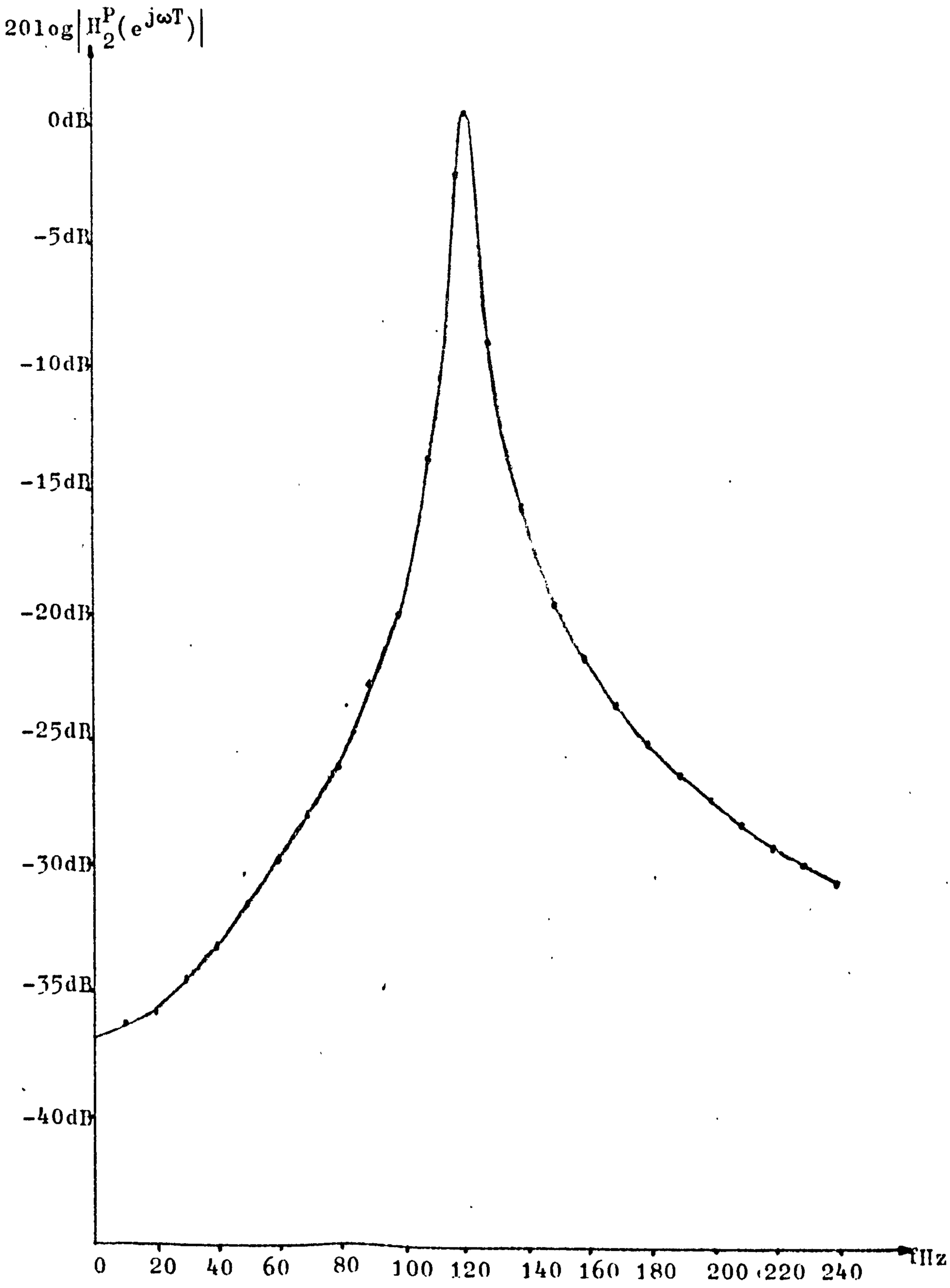


FIG: 8.3.8 FREQUENCY RESPONSE OF THE SECOND PASSBAND ELEMENTAL FILTER $H_2^P(z)$. (WORDLENGTH: INFINITE)

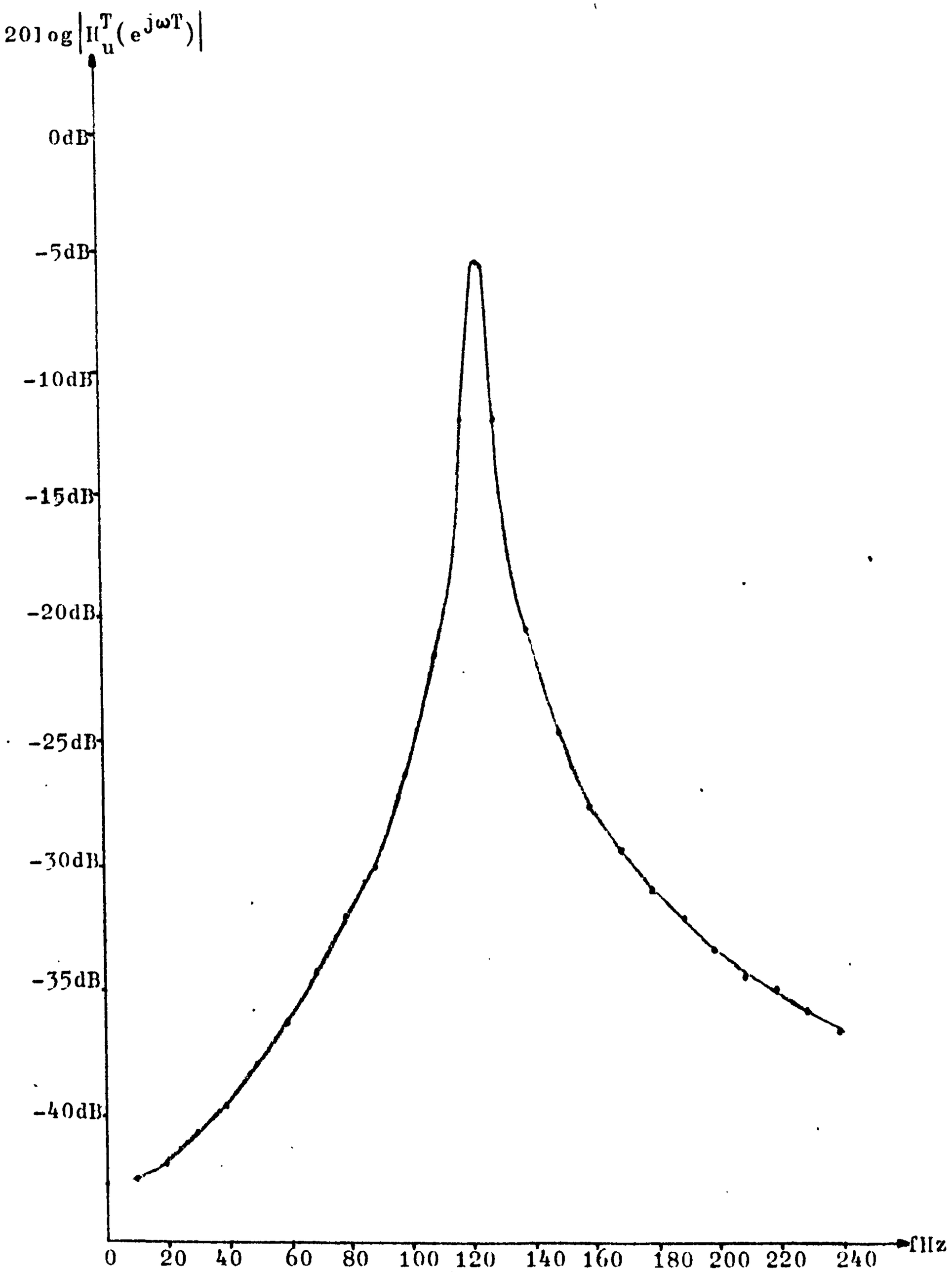


FIG: 8.3.9 FREQUENCY RESPONSE OF UPPER TRANSITION ELEMENTAL FILTER $H_u^T(z)$, $W_n = 0.5$ (WORDLENGTH: INFINITE)

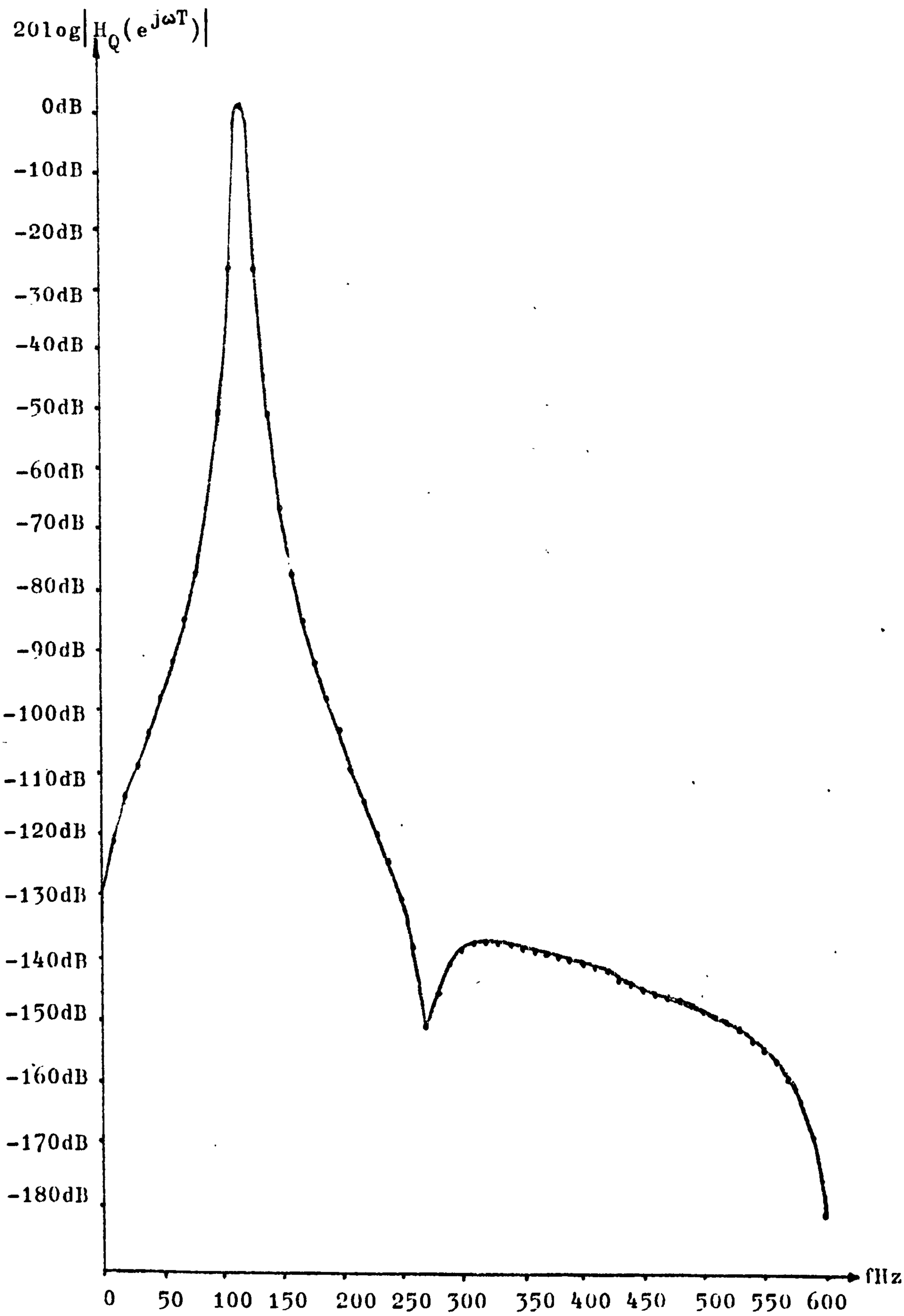


FIG: 8.3.10 FREQUENCY RESPONSE OF A BANDPASS FILTER CENTRED AT 120Hz (WORDLENGTH: INFINITE).

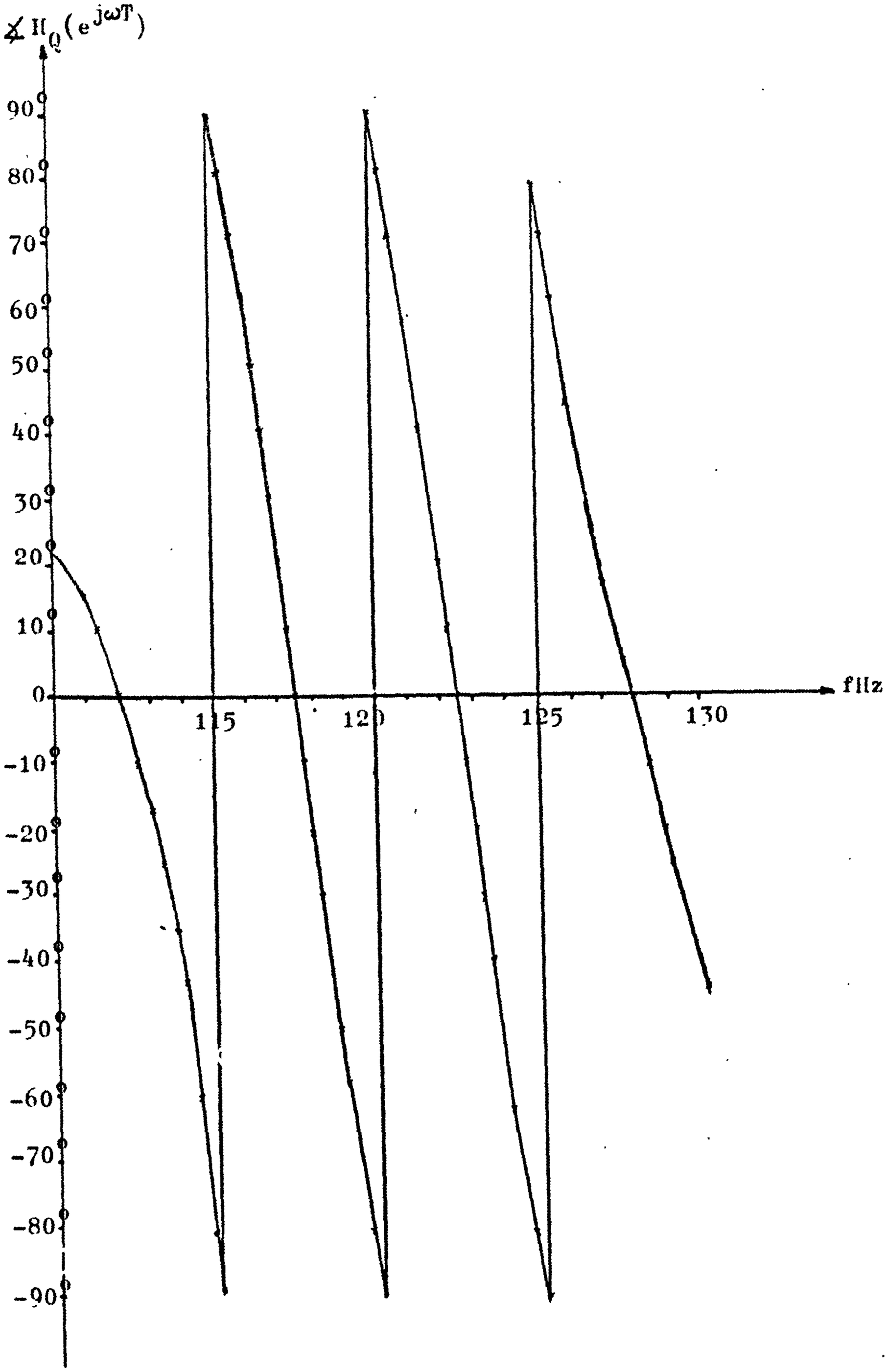


FIG: 8.3.11 PHASE RESPONSE OF A BANDPASS FILTER CENTRED AT 120Hz. (WORDLENGTH: INFINITE).

an anti-symmetry which indicates a piecewise linear phase response. In general, there will be a high degree of piecewise phase linearity in the passband, except at the bandedges, for filters designed by the proposed design technique discussed in chapter three. The non-linearities at the bandedges are due to the optimzation of the transition poles which are seperated from the passband poles at a closer frequency interval than that between passband poles. In fact, it is seen from figure 8.3.11 that if the passband is divided into two sections, from 115Hz to 120Hz and from 120Hz to 125Hz, the phase response within these regions is approximately linear; hence the name piecewise linear. In the next section, we shall analyze the finite word length effects on the frequency response of the above bandpass filter A.

8.3.3 ANALYSIS OF FINITE WORD LENGTH EFFECTS PERTINENT TO SECTION 8.3.1:

Filters implemented by the proposed CBFCS approach discussed in chapter five have been thoroughly analyzed with regards to the effects of finite word lengths, using the proposed generalized quantization noise model discussed in chapter six. In this section, we shall use the results obtained in chapter six for the noise analysis of bandpass filter A.

In section 6.6, it is seen from figure 6.6.1 that the output roundoff error due to product quantization only passes through the poles of the second-order section. This has also been experimentally verified in section 8.2.1 by the fact that the maximum amplitudes of the limit cycles of the transition elemental filters of bandpass filter A are less than those of bandpass filter B. (In configuration (a) shown in figure 8.1.2, the output error due to product quantization is not scaled by the transition sample W_n=0.5 since it does not pass through the zeros of the transition elemental filters.)

Since bandpass filter A has been realized in the parallel form, that is, by parallelling the four constituent elemental filters, then by equation (6.6.21), the output noise variance of bandpass filter A is given by:

$$V(e_{p,n}) = \sum_{i=1}^{4} V(e_{n,i}) \qquad (8.3.1)$$

where $V(e_{n,i})$ is the variance of the ith elemental filter, and can be found by applying equation (6.6.15) while taking into account, the scaling effects of the transition sample W_n=0.5. Furthermore, in equation (8.3.1) above, the summation of the individual output noise variances has been performed under the implicit assumption that these output errors are mutually independent of each other, that is, uncorrelated. Since the value of W_n=0.5, the extra error due to output scaling can be neglected by retaining an extra bit in the summation of the outputs of the four elemental filters of bandpass filter A shown in figure 8.1.2(a). In addition, it was mentioned in section 8.3.1 that the transfer functions of the elemental filters of bandpass filter A had been scaled to prevent signal overflows.

From equation (6.6.15), the output error variance of $H_1^T(z)$ is given by:

$$V(e_{n,1}) = \frac{2^{-16}}{3}\cdot\frac{1}{2\pi j}\oint_{|z|=1} H_1^T(z).H_1^T(z^{-1}).\frac{dz}{z}$$

$$+ \frac{2^{-16}}{3}(1-2^{-30})\frac{1}{2\pi j}\oint_{|z|=1} \frac{(W_n)^2}{P_1^T(z).P_1^T(z^{-1})}\cdot\frac{dz}{z} \qquad (8.3.2)$$

where $P_1^T(z)$ is the pole expression of $H_1^T(z)$, and the values of M and N in equation (6.6.15) are 8 and 16 respectively. Evaluating the integrals in equation (8.3.2) by Cauchy's Residue Theorem, we have

$$V(e_{n,1}) = \frac{2^{-16}}{3}\left[\frac{1+e^{-2aT}}{1-e^{-2aT}}\cdot\frac{S^2}{e^{-4aT}+1-2e^{-2aT}\cos 2(\omega_\phi-b)T}\right](W_n)^2$$

$$+ \frac{2^{-16}}{3}(1-2^{-30})\left\{\frac{1}{1-e^{-2aT}} - e^{-2aT}\sin^2((\omega_\phi-b)T)\right.$$

$$\left.\left[\frac{1+e^{-2aT}}{1-e^{-2aT}}\cdot\frac{1}{e^{-4aT}+1-2e^{-2aT}\cos 2(\omega_\phi-b)T}\right]\right\}(W_n)^2 \qquad (8.3.3)$$

where S is a scaling factor for all elemental filters.

Since the elemental filters of bandpass filter A (and most filters designed by the proposed design technique discussed in chapter three) have poles closed to the unit circle, along which the integrals in equation (8.3.2) are evaluated, we can let $e^{-aT} = 1-\sigma$ and ignore terms with quadratic and higher exponents in σ. Thus, we can rewrite equation (8.3.3) as:

$$V(e_{n,1}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma}(W_n)^2 + \frac{2^{-16}}{3}(1-2^{-30})\left[\frac{1}{4\sigma\sin^2((\omega_p-b)T)}\right](W_n)^2 \qquad (8.3.4)$$

Inspection of the above equation shows that the roundoff noise, which passes through the poles only, is proportional to the resonance angle $(\omega_p-b)T$, while the total output noise variance $V(e_{n,1})$ is proportional to the distance of the poles of $H_1^T(z)$ from the unit circle. These comments also apply to the rest of the elemental filters whose noise variances at their respective outputs can be written similarly as:

$$V(e_{n,2}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma} + \frac{2^{-16}}{3}(1-2^{-30})\frac{1}{4\sigma\sin^2(\omega_p T)} \qquad (8.3.5)$$

$$V(e_{n,3}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma} + \frac{2^{-16}}{3}(1-2^{-30})\frac{1}{4\sigma\sin^2(\omega_p T)} \qquad (8.3.6)$$

$$V(e_{n,4}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma}(W_n)^2 + \frac{2^{-16}}{3}(1-2^{-30})\left[\frac{1}{4\sigma\sin^2((\omega_p+b)T)}\right](W_n)^2 \qquad (8.3.7)$$

Evaluating equations (8.3.4) to (8.3.7) we have the output error of $H_1^T(z)$ being equal to -41.18dB approximately, the output error of $H_1^P(z)$ being equal to -35.32dB approximately, the output error of $H_2^P(z)$ being equal to -35.64dB approximately and, finally, the output error of $H_u^T(z)$ being equal to -41.8dB approximately. These are thus the respective root-mean-square noise outputs which contaminate the frequency responses of the above four elemental filters as shown in the last section in figures 8.3.1 to 8.3.4 which were compared with the ideal frequency responses shown in figures 8.3.6 to 8.3.9.

Finally, by equation (8.3.1), and substituting the above values of the noise variances in equations (8.3.4) to (8.3.7), we have the root-mean-square noise output of the eighth-order bandpass filter A being equal to -31.5dB approximately. This, therefore, explains the discrepancy between the frequency responses of bandpass filter A implemented with eight bits including sign and infinite accuracy respectively shown in figures 8.3.5 and 8.3.10 in the last section. As mentioned above, the roundoff quantization noise of bandpass filter A passes through the poles of its constituent elemental filters only, so that it is proportional to the sampling frequency as can be seen from equations (8.3.4) to (8.3.7). In the design of bandpass filter A, the sampling frequency was chosen to be ten times the centre frequency of the passband of bandpass filter A so that the sines of the resonance angles of the respective elemental filters have values less than unity as can be seen from equations (8.3.4) to (8.3.7). This implies that the output noise due to product roundoff of bandpass filter A can be decreased by a corresponding decrease in the sampling frequency. Nevertheless, the above results obtained so far have further verified the proposed generalized quantization noise model discussed in chapter six for the error analysis of filters implemented by the proposed CBFCS algorithm discussed in chapter five.

8.3.4 SIMULATED FREQUENCY RESPONSES OF A HIGH-FREQUENCY BANDPASS FILTER:

In this section, we present simulation results for the 1MHz bandpass design in section 8.2.4. As explained in section 8.2.4 and section 8.2.5, the above design would require a word length of

well over 16-bit accuracy for its implementation with a tolerable limit cycle behaviour. Furthermore, analogue-to-digital conversions at the rate of the chosen sampling frequency of 3MHz was not possible with the bipolar analogue-to-digital converter (A/D) used, as explained in section 7.15 in chapter seven previously. (In addition, analogue-to-digital conversions in the video region is far more costly than analogue-to-digital conversions in the audio frequency range).

In section 8.2.5, the simulated impulse response of a 1MHz bandpass filter was discussed. Equation (8.2.9) in section 8.2.4 represents the unquantized transfer function H(z) which was then quantized to $H_Q(z)$ in equation (8.2.10), with 16-bit accuracy including sign to represent its coefficients. Figure 8.3.12 shows the frequency responses of H(z) for a few values of the transition sample W_n. As mentioned in section 8.2.4, the optimized value of W_n for the steepest transition skirt is 0.46, which is shown in figure 8.3.12a. It is seen from figures 8.3.12b and 8.3.12c that a further increase in the value of W_n would bring about a larger out-of-band rejection. This, on the other hand, would cause an increased, or wider transition band. Hence, in general, there is a tradeoff between the amount of out-of-band rejection and the width of the transition bands on both sides of the passband (and hence, the steepness of the transition skirt). Furthermore, the value of the in-band ripple ranges from approximately 0.13dB when W_n=0.46 to 0.2dB when W_n=0.49. This then clearly shows the effect of optimization on the in-band ripple magnitude.

Figure 8.3.13 shows the frequency response of $H_Q(z)$ which is seen to be approximating figure 8.3.12a closely. The effect of coefficient quantization has changed the value of the radius of the poles of the elemental filters of the above bandpass filter from 0.9999476 to 0.9999542. Hence, in general, the poles of a digital filter are being moved towards the unit circle due to coefficient quantization. Also, in the process of quantizing H(z) to $H_Q(z)$, the value of W_n has been changed to 0.453125 instead. Figure 8.3.14 shows the phase response of $H_Q(z)$ which is seen to exhibit a high degree of piecewise phase linearity within the passband. The non-linearity at the bandedges is due to the optimization of the transition poles. Such observed characteristics are well expected from the

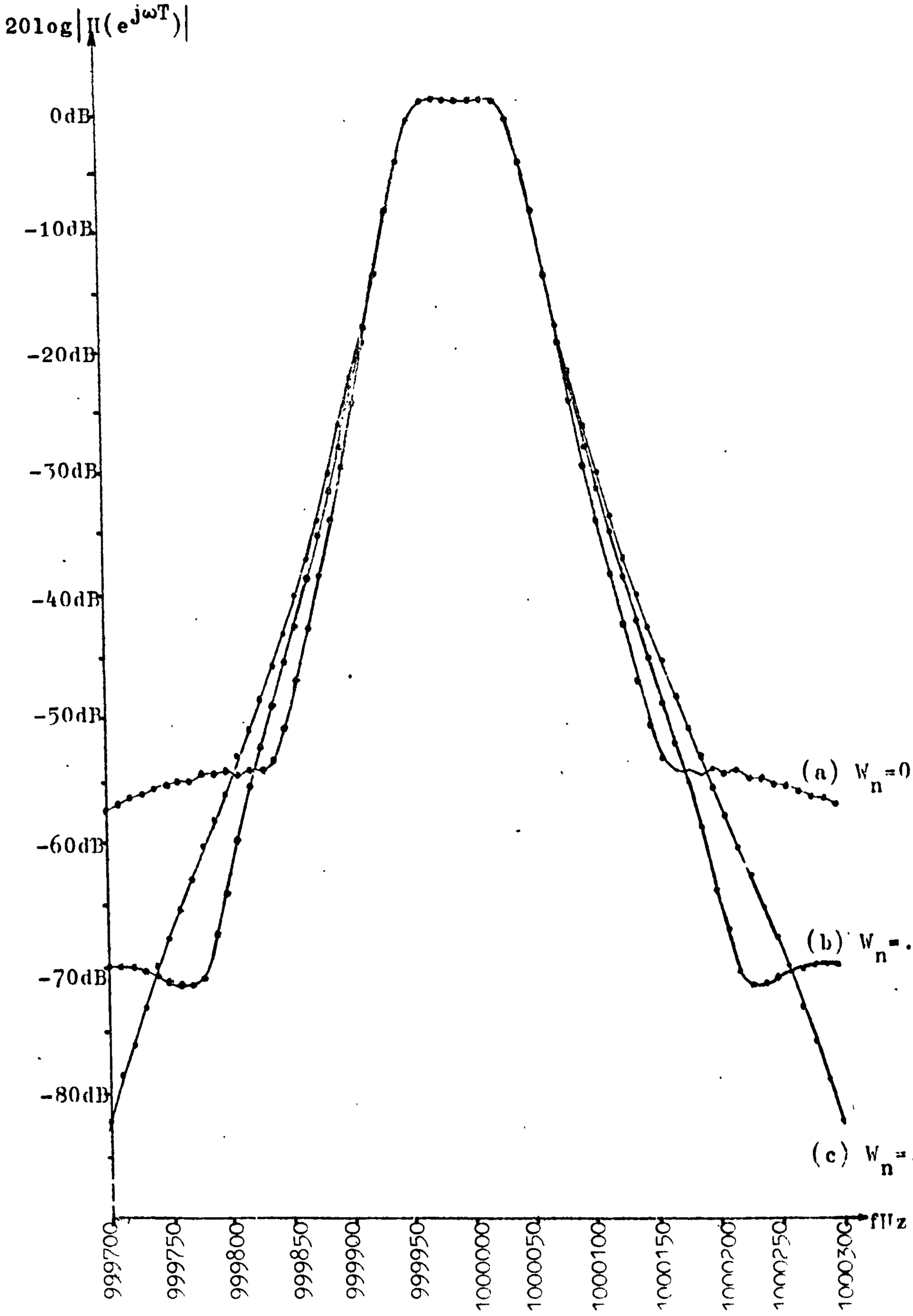


FIG: 8.3.12 FREQUENCY RESPONSES OF A BANDPASS FILTER (1MHz) FOR VARIOUS VALUES OF TRANSITION SAMPLE W_n.

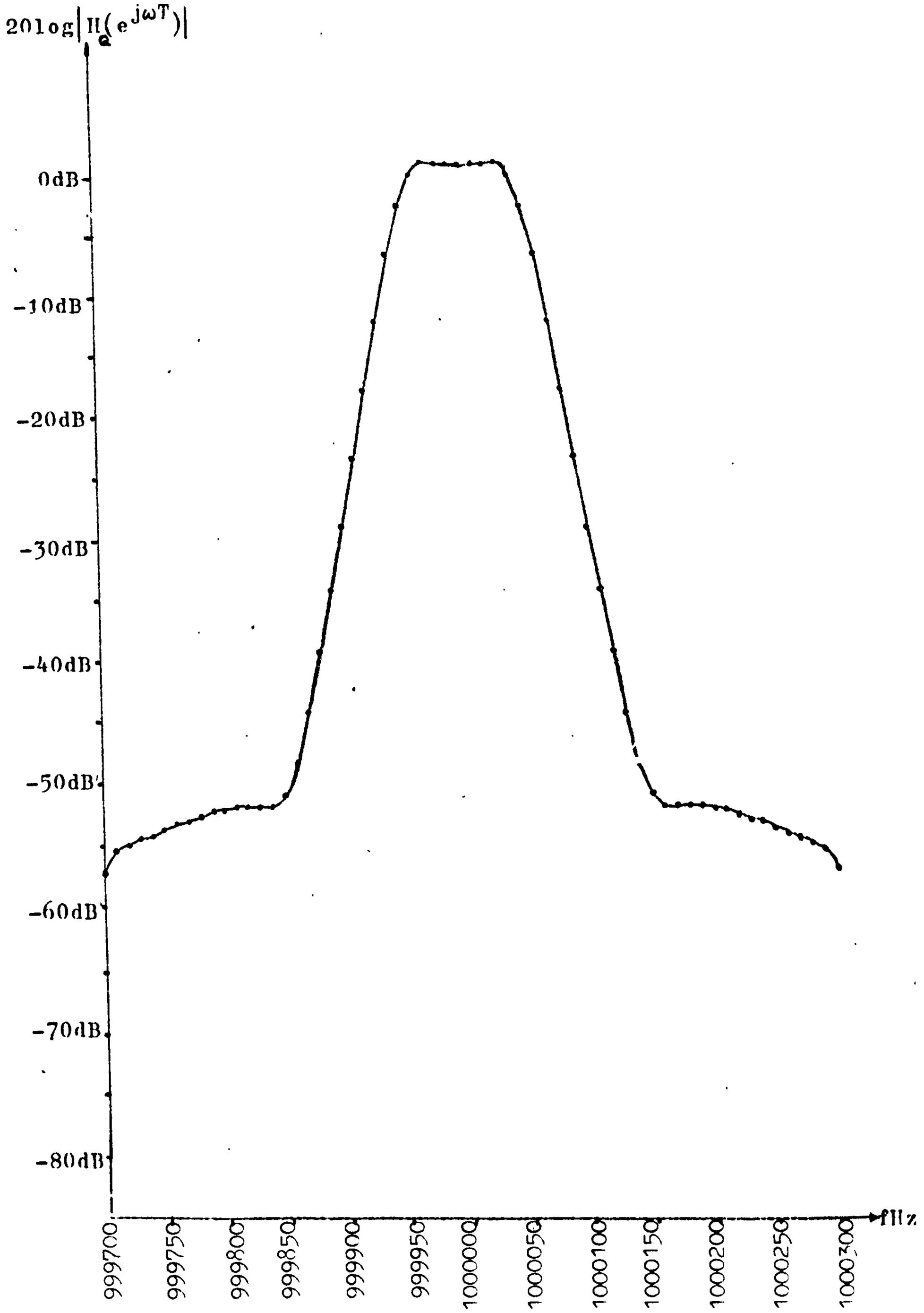
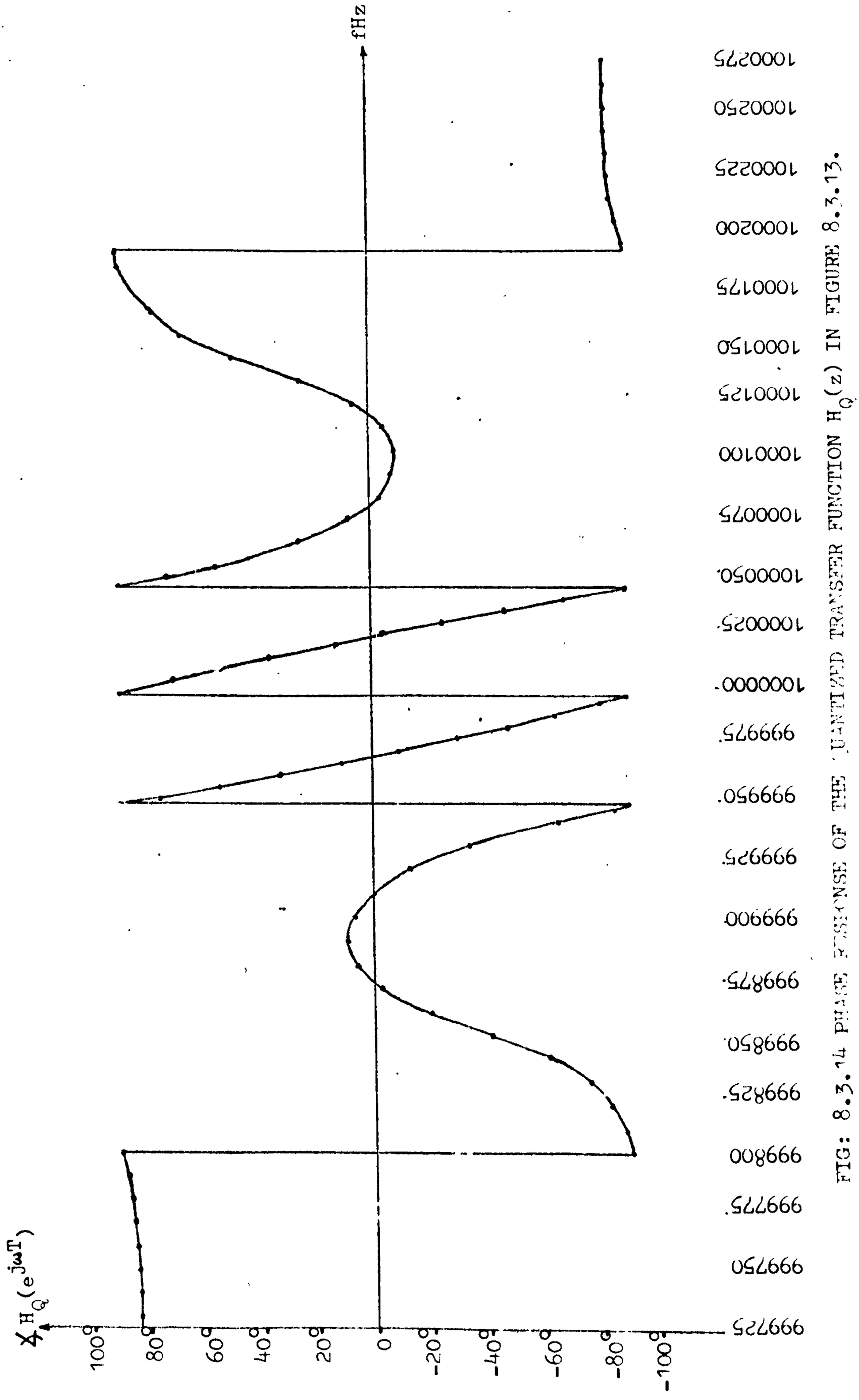


FIG: 8.3.13 FREQUENCY RESPONSE OF A QUANTIZED BANDPASS FILTER (1MHz) WITH OPTIMIZED W_n=0.453125.



8.3.14 PHASE RESPONSE OF THE QUANTIZED TRANSFER FUNCTION $H_Q(z)$ IN FIGURE 8.3.13.

FIG: 8

simulated impulse response of $H_Q(z)$ shown in section 8.2.5 which exhibits an anti-symmetry, except for a small amount of ringing, the effect of which is to slightly impair the phase linearity within the passband. In the above optimized frequency response of $H_Q(z)$ for the steepest transition skirt, it is further noticed that an out-of-band rejection of slightly over -50dB has been achieved at 100Hz immediately outside the passband. (The resultant order of $H_Q(z)$ is eight, while higher-order filters can achieve more out-of-band rejection for the same steepness of the transition skirt).

Finally, to conclude this section, we consider the realization of H(z) via the zero-sharing configuration shown below:

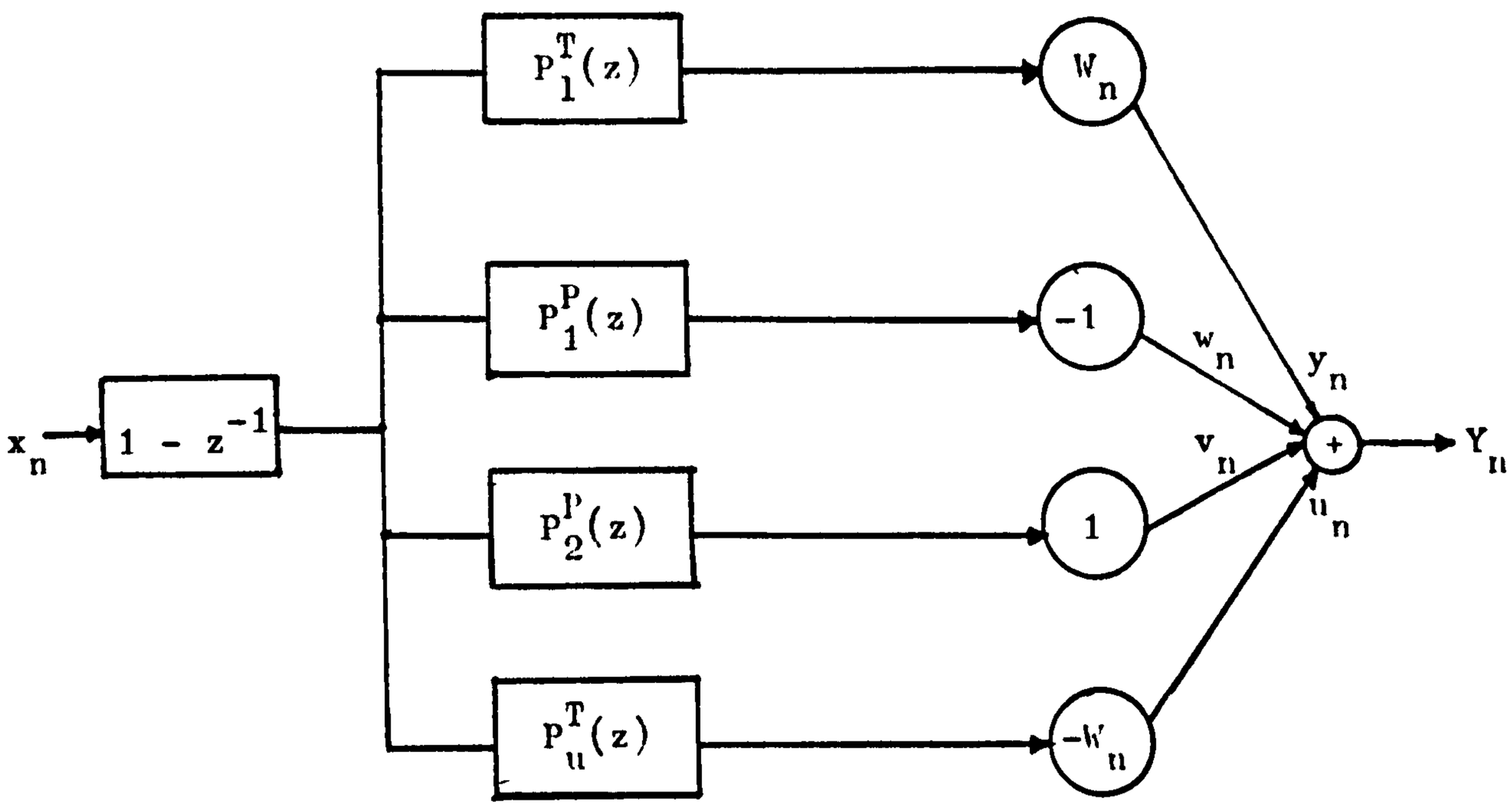
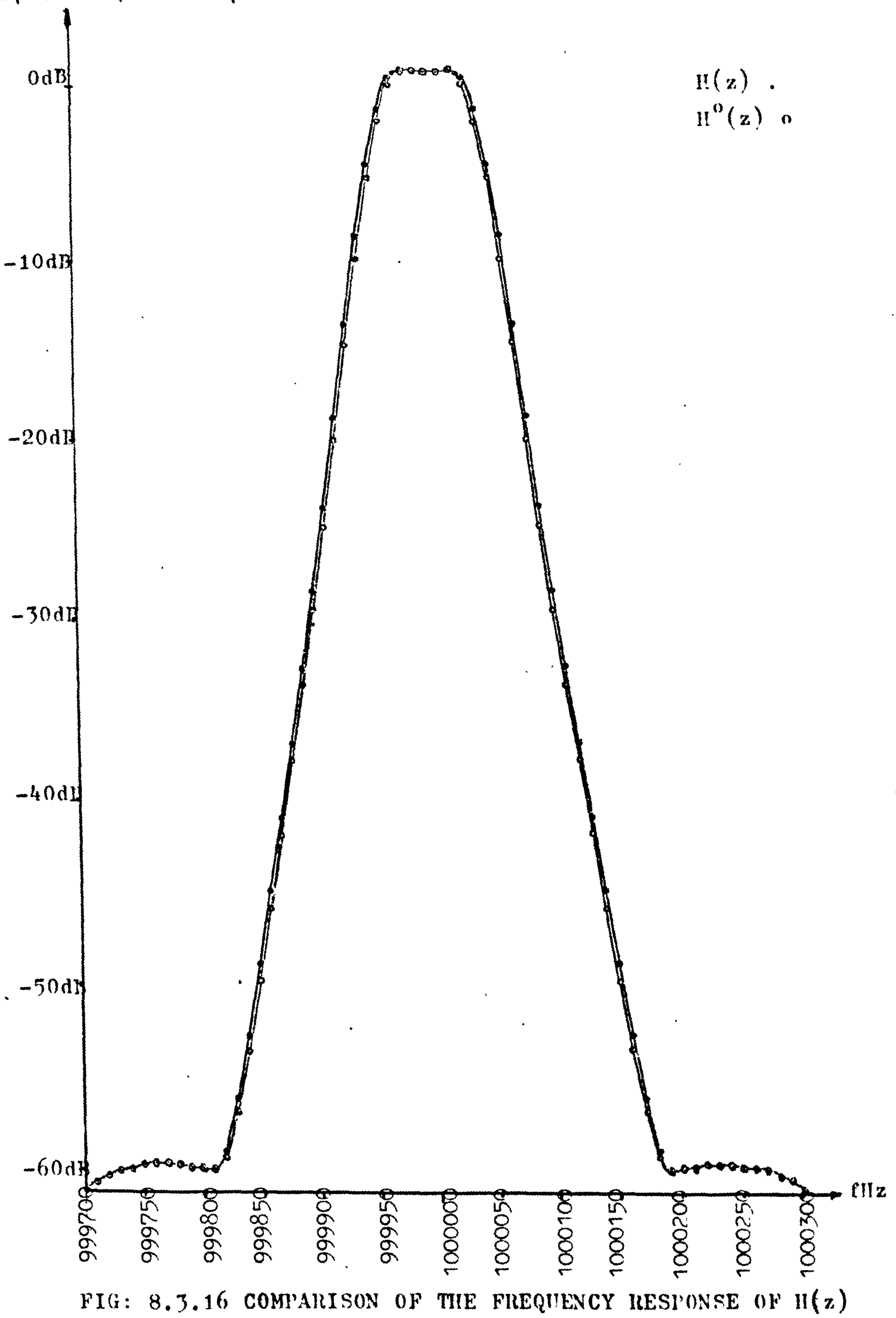


FIG: 8.3.15 ZERO-SHARING REALIZATION OF H(z) IN EQUATION 8.2.9.

The zero-sharing property of the proposed design technique has previously been discussed. In the above figure, $P_1^T(z)$, $P_1^P(z)$, $P_2^P(z)$ and $P_u^T(z)$ are respectively the pole expressions of the elemental filters $H_1^T(z)$, $H_1^P(z)$, $H_2^P(z)$ and $H_u^T(z)$. The term $(1-z^{-1})$ in the above diagram, actually represents the differential input signal $x_n - x_{n-1}$. Figure 8.3.16 compares the frequency response of H(z) and that of its zero-sharing equivalent transfer function $H^0(z)$. It is concluded from figure 8.3.16 that both structures have nearly the same frequency

$20\log|H(e^{j\omega T})|$, $20\log|H^0(e^{j\omega T})|$



$H(z)$.
$H^0(z)$ o

FIG: 8.3.16 COMPARISON OF THE FREQUENCY RESPONSE OF $H(z)$ AND THAT OF ITS ZERO-SHARING EQUIVALENT $H^0(z)$.

response characteristics. (In obtaining the above simulated frequency responses of $H(z)$ and $H^0(z)$, both transfer functions have been normalized and the value of W_n has been chosen arbitrarily as 0.47).

8.4 CONCLUSION:

In this chapter, we have presented experimental and simulation results of filters designed by the proposed design technique discussed in chapter three. A satisfactory agreement between theory and practice has been obtained. The feasibility of the above proposed design technique has also been studied in relation to the choice of word lengths. Furthermore, the experimental results obtained so far have verified the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) for the hardware implementation of digital filters, since they agreed with exact simulation results using the same word length as that of the demonstration processor. The design of the above demonstration processor has also been proven by the same token. Finally, analyses of the above results obtained so far, have further verified the proposed generalized quantization noise model. The proposed "Selective Rounding Scheme" has also been experimentally verified. In the next chapter, we shall consider some suggestions for future research and applications of the proposed design technique and hardware implementation of digital filters for high-speed filtering.

CHAPTER NINE

SUGGESTIONS FOR FUTURE WORK AND APPLICATIONS

9.1 SUGGESTIONS FOR FUTURE WORK:

In this section, we suggest future work involving the improvement of the dynamic range of digital filters implemented by the proposed "Carry Brought Forward Compensation Scheme" (CBFCS), and an alternative implementation of a general second-order difference equation using the above algorithm described and discussed in chapter five previously.

For the improvement of the dynamic range of a digital filter, the use of floating-point arithmetic will be appropriate. For instance, the dynamic range of the demonstration processor described in chapter seven would have been increased had floating-point arithmetic been used. So far, there has not been work done on the application of floating-point arithmetic in relation to the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm in the hardware implementation of high-speed digital filters for realtime filtering. However, the application of floating-point arithmetic to modify the existing "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm is relatively straightforward, and is, therefore, suggested here as a piece of useful future work.

In realtime filtering, the use of small word lengths for the coefficients and variables in the internal arithmetic of digital filters is often desirable. For the same number of bits used in the internal arithmetic of a digital filter, floating-point arithmetic provides a substantially larger dynamic range as compared with fixed-point arithmetic. The large dynamic range provided by the use of floating-point arithmetic can, therefore, provide larger out-of-band rejections for filters designed by the proposed design technique discussed in chapter three, using high-Q (high-gain) elemental filters to sample in frequency both in the passbands and transition bands. Moreover, as mentioned in section 6.8, limit cycle behaviours in digital filters employing floating-point arithmetic can often be ignored while overflow oscillations can also be ruled out in general.

We next suggest an alternative implementation of a general second-order difference equation by the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm discussed in chapter five. Consider the transfer function H(z) of a general second-order section represented by equation (5.2.1) which is repeated below as:

$$y_n = a_o.x_n + a_1.x_{n-1} + a_2.x_{n-2} - b_1.y_{n-1} - b_2.y_{n-2}$$

The transfer function is therefore,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=o}^{K=2} a_i.z^{-i}}{1 + \sum_{i=1}^{K=2} b_i.z^{-i}}$$

which can be written as a cascade of two transfer functions $H_1(z)$ and $H_2(z)$ as follows:

$$H(z) = \frac{Y(z)}{X(z)} = \left[\frac{1}{1 + \sum_{i=1}^{K=2} b_i.z^{-i}}\right]\left[\sum_{i=o}^{K=2} a_i.z^{-i}\right]$$

$$= H_1(z).H_2(z) \qquad (9.1.1)$$

where $H_1(z)$ is seen to be a transfer function containing the poles of H(z) only and $H_2(z)$ contains the zeros of H(z) only. If we write

$$H_1(z) = \frac{W(z)}{X(z)} = \frac{1}{1 + \sum_{i=1}^{K=2} b_i.z^{-i}} \qquad (9.1.2)$$

$$H_2(z) = \frac{Y(z)}{W(z)} = \sum_{i=o}^{K=2} a_i.z^{-i} \qquad (9.1.3)$$

we obtain the pair of difference equations:

$$w_n = x_n - b_1 \cdot w_{n-1} - b_2 \cdot w_{n-2} \qquad (9.1.4)$$

$$y_n = a_o \cdot w_n + a_1 \cdot w_{n-1} + a_2 \cdot w_{n-2} \qquad (9.1.5)$$

which are of the second-order and can be similarly realized with the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm using equations (5.2.10) to (5.2.15) in the serial-mode implementation as shown in figure 5.3.3 in chapter five. In particular, equation (9.1.4) can also be realized as shown in figure 5.4.5 since the filter coefficient for the variable x_n is unity.

Since the number of variables in the above equations (9.1.4) and (9.1.5) are less than that of the general second-order difference equation as represented by equation (5.2.1), there is a corresponding reduction in the size of the CBFCS memory required to implement the above equations. This is obvious from equation (5.3.1). In fact, there is nearly a three-fold saving in the CBFCS memory requirement. However, we hasten to add that because of the cascade implementation of equation (9.1.1), two "Adderless-Multiplierless Units" (AMU's) as well as a proportional increase in other associated logic hardware are now required. On the other hand, if the signal being processed is not a wide-band one in comparison with the operational speed of a basic second-order section, the same set of hardware can then be multiplexed to implement the above cascade structure as shown in figures 5.4.1 and 5.4.2 as described in section 5.4 in chapter five.

A notable feature of the above alternative implementation of equation (5.2.1) is the noise performance of the resultant cascade structure. Figure 6.6.1 in chapter six can be modified for equation (9.1.1) as shown in figure 9.1.1 where two output quantizers are now present. It is seen from figure 6.6.1 that the output noise only passes through the poles of the transfer function whereas, in figure 9.1.1, the output noise of $H_1(z)$ also passes through and is filtered by the zeros $H_2(z)$ of the transfer function $H(z)$. Hence, it is seen that the output noise of the realization of the second-order difference equation (5.2.1) represented by figure 6.6.1 will be filtered by the poles of $H(z)$ only while being accentuated by the effects of the poles. On the other hand, the output noise of the first

section, $H_1(z)$, is accentuated by the effects of the poles of $H(z)$, but also attenuated by the effects of the zeros, $H_2(z)$ of the transfer function $H(z)$. Moreover, since $H_2(z)$ is a non-recursive structure, the output noise of the second output quantizer is simply an additive noise at the output of the cascade structure.

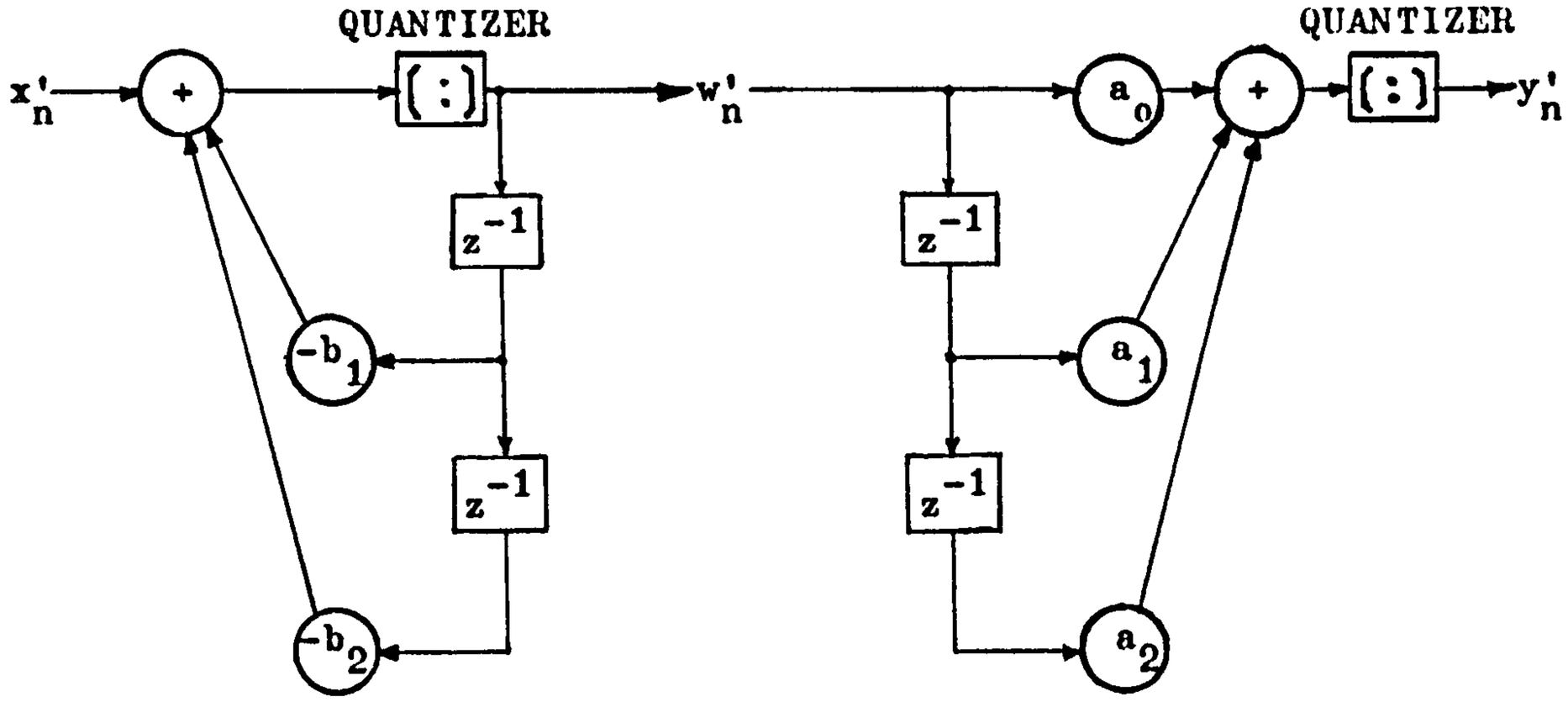


FIG: 9.1.1 BLOCK DIAGRAM OF AN ALTERNATIVE IMPLEMENTATION OF A GENERAL SECOND-ORDER QUANTIZED DIGITAL FILTER BY THE PROPOSED CBFCS ALGORITHM

In the above diagram, x'_n, w'_n and y'_n are respectively x_n, w_n and y_n contaminated with noise. The overall noise expression for the above configuration is:

$$V(e_n) = \frac{2^{-2M}}{3}.\frac{1}{2\pi j}\oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z}$$

$$+ \frac{2^{-2M}}{3}(1-2^{-2(N-1)})\frac{1}{2\pi j}\oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z}$$

$$+ \frac{2^{-2M}}{3}(1-2^{-2(N-1)}) \qquad (9.1.6)$$

where the first term is due to input quantization and the latter two terms are due to product quantization in implementing $H_1(z)$ and $H_2(z)$.

To conclude this section, it will be worthwhile to apply the configuration shown in figure 9.1.1 for an alternative implementation of bandpass filter A discussed in chapter eight. By equation (9.1.6), equations (8.3.4) to (8.3.7) are correspondingly modified under the configuration shown in figure 9.1.1 to become as follows:

$$V(e_{n,1}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma}(W_n)^2 + \frac{2^{-16}}{3}(1-2^{-30})\frac{S^2}{4\sigma}(W_n)^2 + \frac{2^{-16}}{3}(1-2^{-30}) \qquad (9.1.7)$$

$$V(e_{n,2}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma} + \frac{2^{-16}}{3}(1-2^{-30})\frac{S^2}{4\sigma} + \frac{2^{-16}}{3}(1-2^{-30}) \qquad (9.1.8)$$

$$V(e_{n,3}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma} + \frac{2^{-16}}{3}(1-2^{-30})\frac{S^2}{4\sigma} + \frac{2^{-16}}{3}(1-2^{-30}) \qquad (9.1.9)$$

$$V(e_{n,3}) = \frac{2^{-16}}{3}\cdot\frac{S^2}{4\sigma}(W_n)^2 + \frac{2^{-16}}{3}(1-2^{-30})\frac{S^2}{4\sigma}(W_n)^2 + \frac{2^{-16}}{3}(1-2^{-30}) \qquad (9.1.10)$$

which are independent of the respective resonance angles, viz., $(\omega_p-b)T$, $(\omega_p T)$, $(\omega_p T)$ and $(\omega_p+b)T$ of the four elemental filters of bandpass filter A in chapter eight. Hence, the total quantization error at the output of bandpass filter A implemented under the alternative configuration shown in figure 9.1.1 will only be dependent on the distance of the poles of its constituent elemental filters from the unit circle. As explained in section 8.3.3, the total output error due to quantization of bandpass filter A, implemented with the alternative configuration shown in figure 9.1.1 for its constituent elemental filters, has a variance equal to the sum of the variances given in equations (9.1.7) to (9.1.10) above. Nevertheless, the choice of which configuration is to be employed in a particular application also depends on the path of the signal through the filter configuration, that is, its attenuation and amplification.

9.2 SUGGESTIONS FOR APPLICATIONS OF THE PROPOSED DESIGN TECHNIQUE FOR REALTIME DIGITAL FILTERS:

Since the proposed design technique discussed and described in chapter three is particularly suitable for designing filter banks, because of the inherent pole-sharing and zero-sharing properties, we suggest here, in this section, a few possible applications of the above technique where a large number of channels or filters are required.

In addition, it is worth pointing out that since linear phase approximation is feasible with the above technique, and that linear phase filters are important for applications where frequency dispersions are harmful, the above technique will then find useful applications in areas such as speech processing and data transmission etc.

9.2.1 APPLICATION IN DIGITAL SPECTRUM ANALYSIS:

Figure 9.2.1 below shows a typical digital spectrum analyzer.

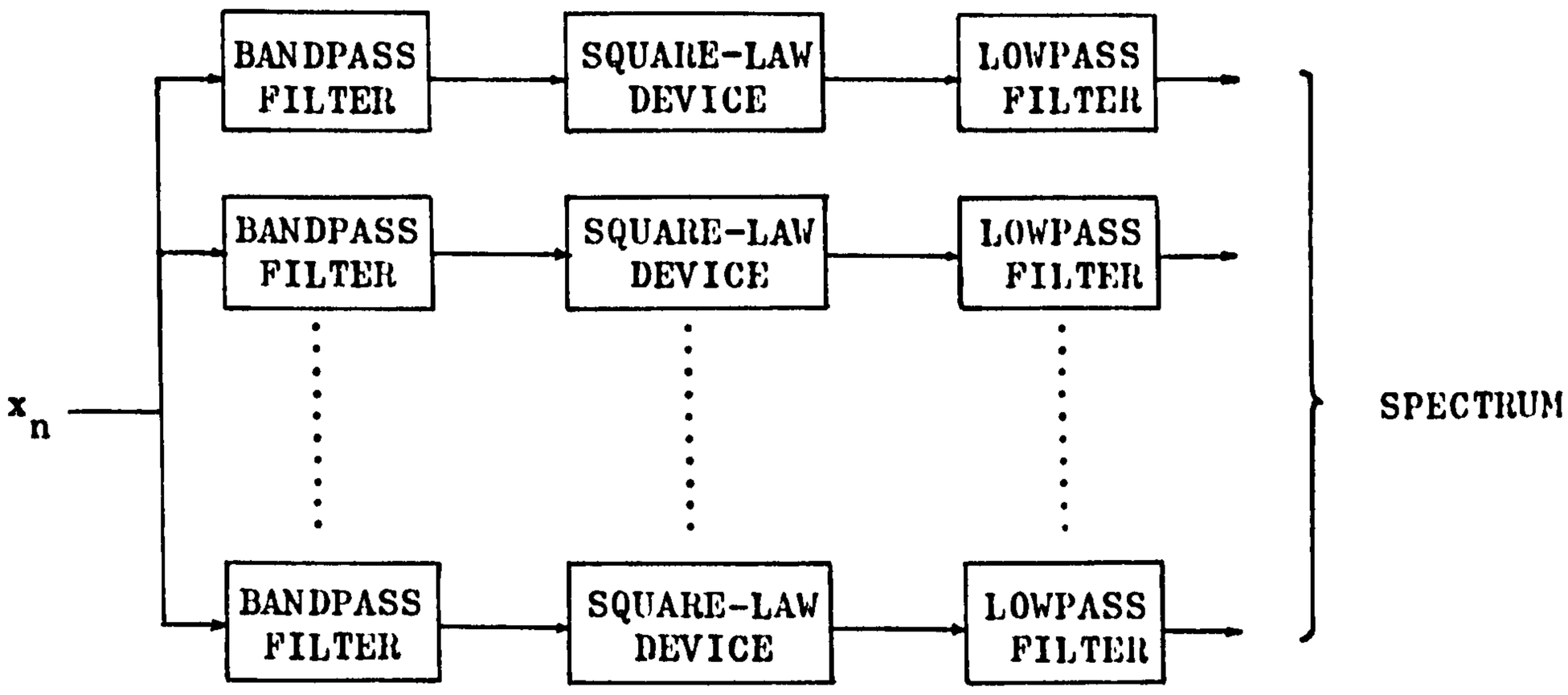


FIG: 9.2.1 BLOCK DIAGRAM OF A TYPICAL DIGITAL SPECTRUM ANALYZER

The energy of the input signal x_n, at a given spectral location, is measured by a bandpass filter in the first filter bank, tuned to

the proper frequency, followed by a square-law device and a lowpass filter in the second filter bank to smooth the resultant power measurement. These lowpass filters can be realized with equation (3.2.29) as described in chapter three.

In general, the individual filters in the first bank of bandpass filters are contiguous such that some of the elemental filters used to implement them can be shared according to the inherent pole-sharing and zero-sharing properties of the proposed design technique as discussed in sections 3.3 and 3.4. In addition, section 3.6 expands on the possibility of realtime calculations of the required filter coefficients via an interpolation formula. Finally, section 3.5 comments on some of the conventional schemes for the realization of the above filters.

9.2.2 APPLICATION IN CHANNEL VOCODERS:

The channel vocoder (voice coder) is an analysis-synthesis system based on our knowledge of the speech production and perception mechanism in man. Figure 9.2.2 shows a block diagram of a typical channel vocoder. The incoming speech x_n is analyzed by a bank of bandpass filters that non-uniformly cover the frequency range of interest (generally 0 to 3kHz). The outputs of these bandpass filters are rectified and filtered by the bank of lowpass filters that followed. The outputs of these lowpass filters then more or less represent the spectral envelope of the speech signal. Parameters representing the nature of the excitation are obtained by the voiced-unvoiced detector that determines if the speech is voiced (i.e., the vocal cords are vibrating) or unvoiced. If the speech is voiced, a pitch extractor determines the fundamental frequency of vibration, F_o.

The outputs of the bank of lowpass filters and the voiced-unvoiced and fundamental frequency signals are coded and transmitted over a channel to the receiver. Assuming error-free transmission, the job of the receiver is to reconstruct the speech, that is, to synthesize speech from the transmitted parameters. The excitation is created from either a pulse generator whose fundamental frequency is controlled by the F_o signal or a random noise generator. The voiced-unvoiced signal controls a switch to determine which source will

excite a bank of bandpass filters that are identical to those used in the analysis section. The lowpass spectrum envelope signals modulate the outputs of the bandpass filters to control the speech power in each of the frequency bands. The resultant synthetic speech is obtained by summing the modulated outputs of the bank of bandpass filters in the synthesis section. One of the goals of the channel vocoder is low bit-rate transmission of reasonable quality speech.

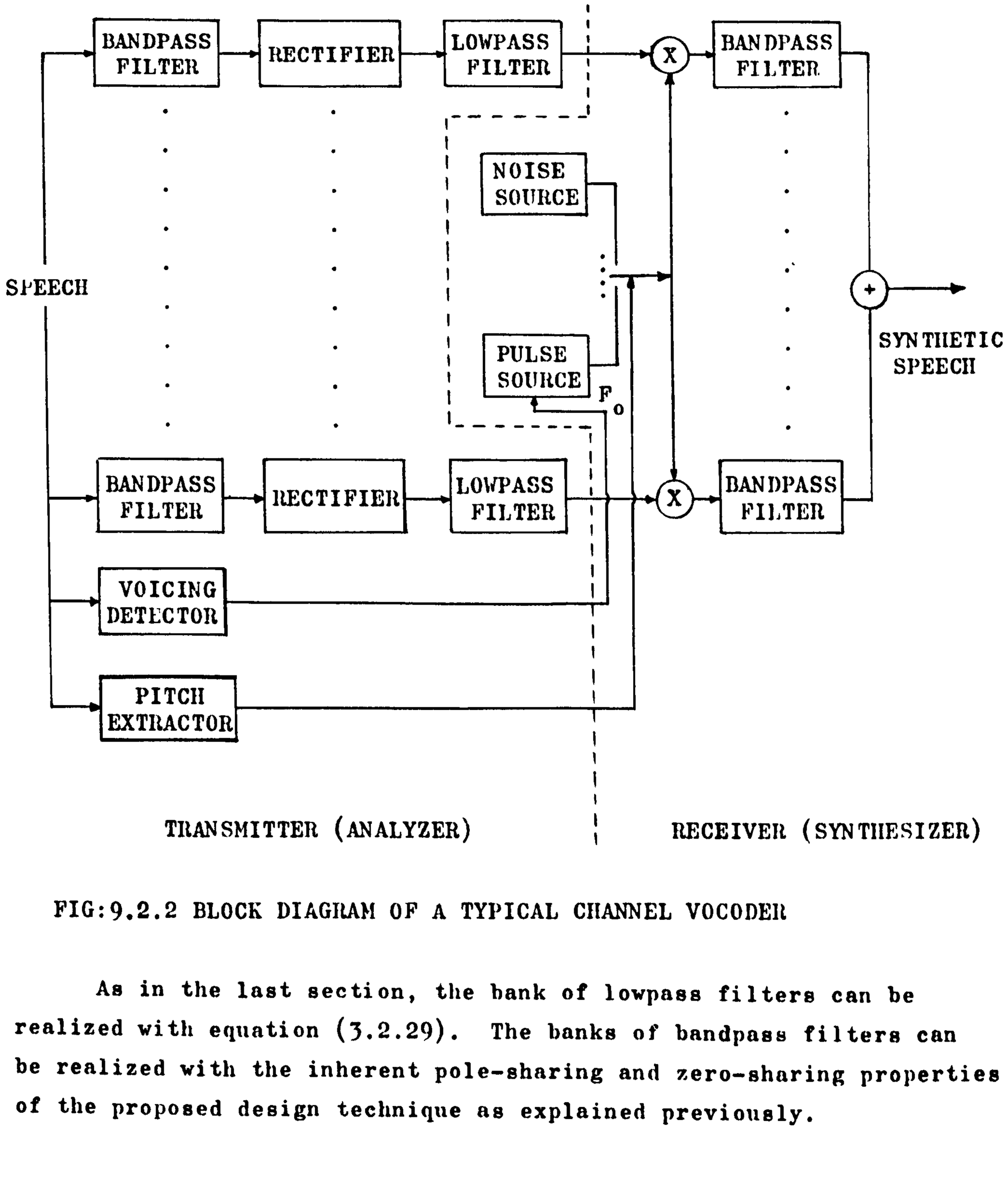


FIG:9.2.2 BLOCK DIAGRAM OF A TYPICAL CHANNEL VOCODER

As in the last section, the bank of lowpass filters can be realized with equation (3.2.29). The banks of bandpass filters can be realized with the inherent pole-sharing and zero-sharing properties of the proposed design technique as explained previously.

9.2.3 APPLICATION IN FM-CW RADAR SIGNAL PROCESSING:

Figure 9.2.3 below shows a simplified block diagram of a frequency-modulnted continuous-wave (FM-CW) radar.

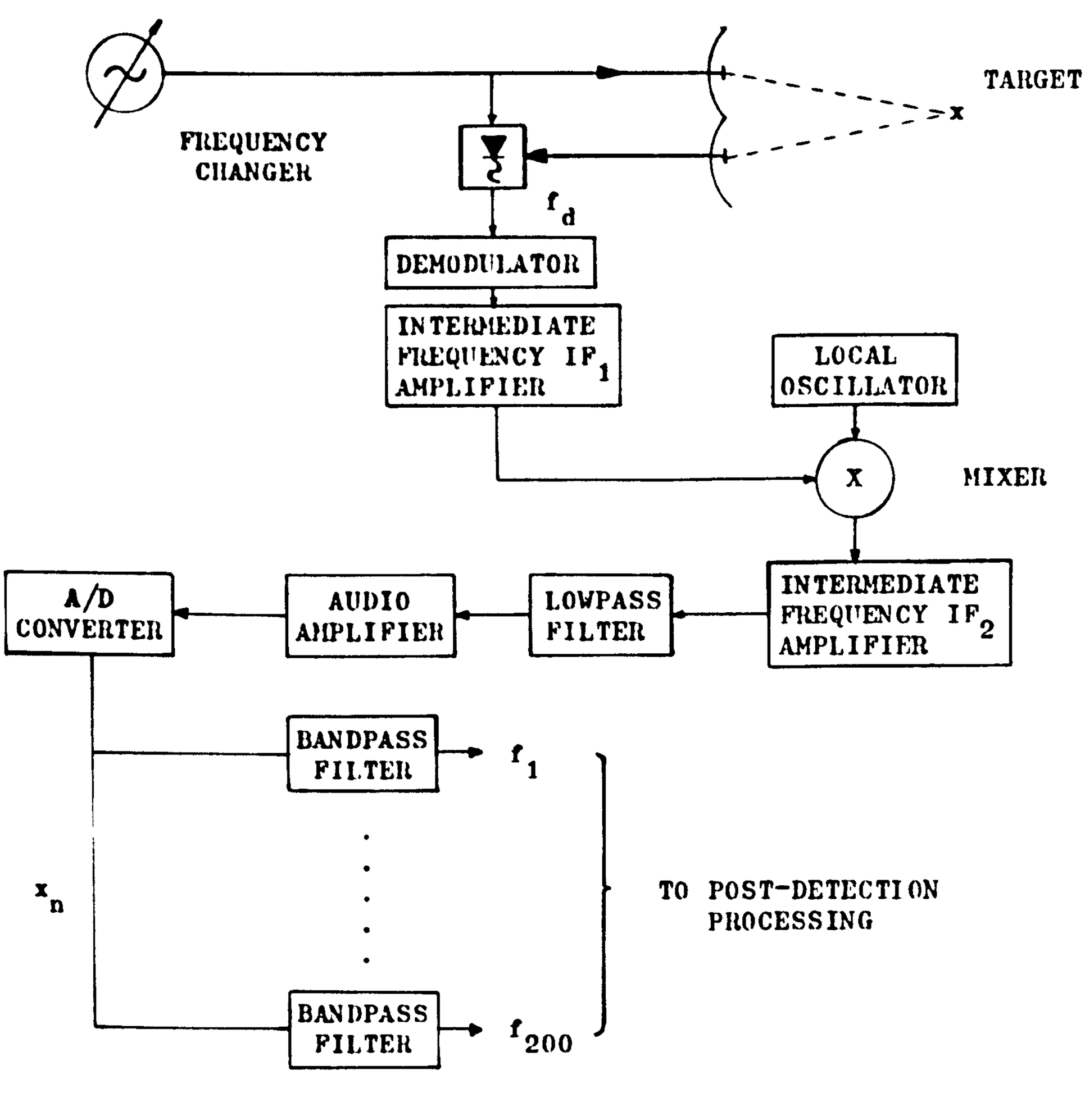


FIG: 9.2.3 A SIMPLIFIED BLOCK DIAGRAM OF A FREQUENCY-MODULATED CONTINUOUS-WAVE (FM-CW) RADAR

In the above FM-CW radar system, the bank of two hundred bandpass filters are used to filter the beat frequency f_d due to the

beating of the transmitted and the return signals. When the bandwidth covered by the filter bank of bandpass filters is within the range of 0 to 3kHz, the processing of the signal returns from possible targets can then be done in the audio range. This enhances the application of digital filtering since the arithmetic unit of the filtering unit can then be multiplexed to implement the filter bank. The signal f_d is first demodulated and then translated in frequency from the first intermediate frequency IF_1 by the local oscillator and the mixer down to the second intermediate frequency IF_2 which is in the audio range.

After lowpass filtering and audio amplification, the above signal is then converted into a digital signal via the analogue-to-digital converter and then fed into the bank of digital bandpass filters which can conveniently be designed via the proposed design technique in chapter three. As in the former applications, due to the pole-sharing and zero-sharing properties of the above proposed design technique, the resultant filter bank is both computationally efficient and economical which are important in realtime filtering. Owing to the typically large number of signal returns per rotation of the antennae, some two hundred bandpass filters are often required to process these signal returns. These bandpass filters have equal bandwidth and are contiguous, centred at the frequencies of the possible signal returns. The outputs of these bandpass filters are often passed on to some post-detection processing for display or other decision-making purposes. An advantage of processing in the audio range is the relatively low cost in analogue-to-digital conversion. A disadvantage of processing in the audio range is the limitation in range of the resultant radar system, since a relatively narrow-band spectrum of signal returns can be detected or accommodated.

9.3 SUGGESTIONS FOR APPLICATIONS OF THE PROPOSED "CARRY BROUGHT FORWARD COMPENSATION SCHEME" (CBFCS) ALGORITHM:

In essence, the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm proposed in chapter five is a high-speed computational mechanism that is applicable in most digital signal processors where multiplications and additions are performed. In particular, the above algorithm can also be applied to modify the arithmetic units of present day "Microprocessors" since the above

algorithm employs mainly memory in its hardware and is very suitable for large scale integration. The "Adderless-Multiplierless Unit" (AMU) in conjunction with the required CBFCS memory discussed in chapter five can also directly replace the multipliers and adders for multiplications and additions of most existing hardware arithmetic units, substituting table lookup for computation.

As previously explained in chapter five, "psuedo-pipeline" structures result when the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm is used to implement recursive difference equations. On the other hand, when non-recursive difference equations are being realized with the above proposed algorithm, then genuine pipeline structures result. In view of this, the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm is particularly adapted to pipeline structures for the case of non-recursive realization. Examples are pipeline convolvers for high-speed convolutions, and pipeline Fast Fourier Transformers(FFT). In the follow section, we shall briefly describe the application of the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm in a high-speed Fast Fourier Transformer. Due to the novelty of the above proposed algorithm, the following implementation of a high-speed Fast Fourier Transformer is also believed to be novel.

9.3.1 APPLICATION IN THE HARDWARE IMPLEMENTATION OF A FAST FOURIER TRANSFORMER:

In order to achieve a high throughput rate in a special-purpose Fast Fourier Transform (FFT) processor, conventional implementations require a large number of multipliers. This is often associated with a high power consumption and may be the limiting factor in some applications. In this section, we propose a high-speed hardware implementation of a Fast Fourier Transformer using the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm with a modified floating-point arithmetic to permit a wider dynamic range.

The Fast Fourier Transform (FFT) computes the discrete Fourier Transform of a sequence of N input numbers $\{x_n\}$ where $0 \leqslant n \leqslant N-1$, such that

$$X_l = \sum_{n=0}^{N-1} x_n . W^{nl} \qquad l = 0, 1, \ldots, N-1 \qquad (9.3.1)$$

where W = exp(-2πj/N) with j =$\sqrt{-1}$. The FFT algorithms factor N into a product of integers and compute the final transform throught repeated application of a lower dimension transform, gaining considerable advantage in speed and storage requirement. In this section, we consider specifically the radix-2 decimation-in-time (132) algorithm with $N=2^L$, L integral.

The basic operation of the decimation-in-time algorithm is the so-called butterfly (shown in figure 9.3.1) in which the input tuple (A, B, k) are combined to give the output pair (X, Y) via the operations:

$$X = A + B.W^k \tag{9.3.2}$$

$$Y = A - B.W^k \tag{9.3.3}$$

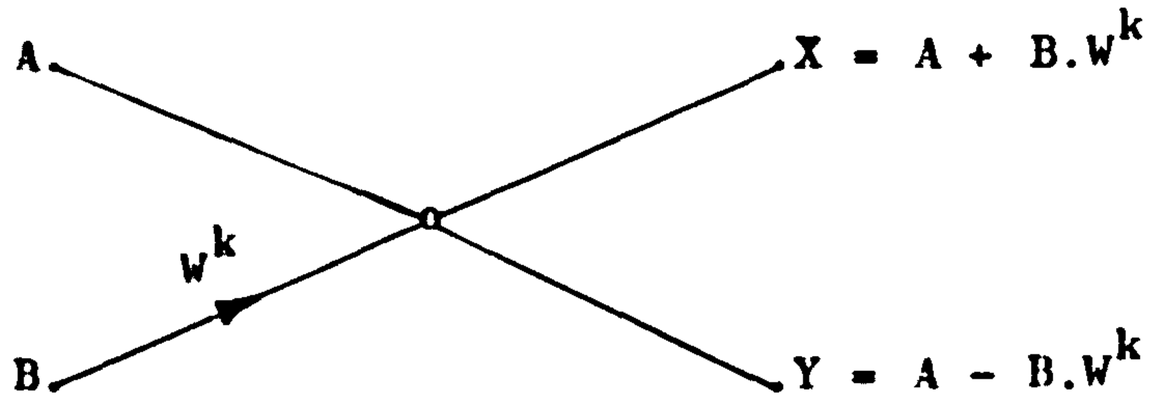


FIG: 9.3.1 THE RADIX-2 DECIMATION-IN-TIME ALGORITHM FFT BUTTERFLY.

In the above equations, k is an integer in the range $0 \leq k < N/2$, depending on how many steps of the algorithm above have been executed and is mostly supplied by the control circuitry in a pipeline FFT processor. The inputs A and B are two complex numbers so that the multiplications in the above equations are complex. A pipeline FFT system is a hardware architecture designed to speed up the data rate throughput for realtime applications (133). Essentially, such a system is characterized by its $L = \log_2 N$ independent arithmetic units operating concurrently to increase the throughtput rate L times. If T is the time required for an arithmetic unit above to compute the output pair (X, Y) according to equations (9.3.2) and (9.3.3), then the complex data throughput rate of such a pipeline FFT processor can be up to 2/T. The number of arithmetic units needed is equal to the

number of stages L = $\log_2 N$, thus 10 arithmetic units are needed for a 1024-point FFT processor. An obvious design goal is to minimize the hardware and power consumption of each arithmetic unit for a given throughput rate.

We next examine the operations of the FFT butterfly in order to implement it with the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm, to replace the multipliers in conventional arithmetic units of a pipeline FFT processor. Let the complex numbers A and B be represented in floating-point format as follows:

$$A = 2^m(a_r + ja_i) \tag{9.3.4}$$

$$B = 2^n(b_r + jb_i) \tag{9.3.5}$$

where the subscripts r and i denote respectively the real and imaginary parts, and the integers m and n are common exponents such that $-1 \leq a_r, a_i, b_r, b_i < 1$. Equations (9.3.2) and (9.3.3) can now be rewritten as:

$$X = 2^m(a_r + ja_i) + Z \tag{9.3.6}$$

$$Y = 2^m(a_r + ja_i) - Z \tag{9.3.7}$$

where

$$Z = 2^n(b_r + jb_i)(\cos \frac{2\pi k}{N} - j\sin \frac{2\pi k}{N}) \tag{9.3.8}$$

Thus, the real and imaginary parts of Z are given by

$$z_r = b_r.\cos \frac{2\pi k}{N} + b_i.\sin \frac{2\pi k}{N} \tag{9.3.9}$$

$$z_i = -b_r.\sin \frac{2\pi k}{N} + b_i.\cos \frac{2\pi k}{N} \tag{9.3.10}$$

To compute X and Y we first compute Z, which involves four real multiplications and two real additions. Then A and Z are scaled jointly and added/subtracted to give X and Y. In conventional systems, to achieve maximum speed, four real multipliers, a scaler and six adders are needed. The throughput rate is then determined largely by the time needed to perform one real multiplication.

Now let us consider an alternative way of computing Z using the serial-mode of the "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm proposed in section 5.2 and let the "byte-sliced" summations of the implementation of the above algorithm be performed in Q-bit bytes. Assuming two's complement code is used, with M bits including sign, to represent the mantissa, then

$$a_r = -a_r^0 + \sum_{j=1}^{M-1} a_r^j . 2^{-j} \qquad a_r^j \in \{0,\ 1\} \qquad (9.3.11)$$

$$a_i = -a_i^0 + \sum_{j=1}^{M-1} a_i^j . 2^{-j} \qquad a_i^j \in \{0,\ 1\} \qquad (9.3.12)$$

$$b_r = -b_r^0 + \sum_{j=1}^{M-1} b_r^j . 2^{-j} \qquad b_r^j \in \{0,\ 1\} \qquad (9.3.13)$$

$$b_i = -b_i^0 + \sum_{j=1}^{M-1} b_i^j . 2^{-j} \qquad b_i^j \in \{0,\ 1\} \qquad (9.3.14)$$

Suppose two's complement code is used, with P bits including sign, to represent the coefficients of the real multiplications in equations (9.3.9) and (9.3.10), then

$$\cos \frac{2\pi k}{N} = p_k = -p_k^0 + \sum_{i=1}^{P-1} p_k^i . 2^{-i} \qquad p_k^i \in \{0,\ 1\} \qquad (9.3.15)$$

$$\sin \frac{2\pi k}{N} = q_k = -q_k^0 + \sum_{i=1}^{P-1} q_k^i . 2^{-i} \qquad q_k^i \in \{0,\ 1\} \qquad (9.3.16)$$

Now we can rewrite equations (9.3.9) and (9.3.10) as follows:

$$z_r = -g_k(b_r^0,\ b_i^0,\ C_Q^1,\ C_{2Q}^1,\ \ldots\ldots,\ C_{\left(\frac{P}{Q}-1\right)Q}^1) + \sum_{j=1}^{M-1} 2^{-j} . g_k(b_r^j,\ b_i^j,\ C_Q^{j+1},\ C_{2Q}^{j+1},\ \ldots\ldots,\ C_{\left(\frac{P}{Q}-1\right)Q}^{j+1}) + g_k' + g_k'' \qquad (9.3.17)$$

$$z_i = -h_k(b_r^0,\ b_i^0,\ C_Q^1,\ C_{2Q}^1,\ \ldots\ldots,\ C_{\left(\frac{P}{Q}-1\right)Q}^1) + \sum_{j=1}^{M-1} 2^{-j} . h_k(b_r^j,\ b_i^j,\ C_Q^{j+1},\ C_{2Q}^{j+1},\ \ldots\ldots,\ C_{\left(\frac{P}{Q}-1\right)Q}^{j+1}) + h_k' + h_k'' \qquad (9.3.18)$$

where g_k and h_k, which provide the necessary "compensated partial products" in the "byte-sliced" summations of the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm, are defined as the function f in equation (5.2.10) in section 5.2 in chapter five, and shown as below:

$$g_k(u^j,\ v^j,\ C_Q^{j+1},\ C_{2Q}^{j+1},\ \ldots\ldots,\ C_{\left[\frac{P}{Q}-1\right]Q}^{j+1})$$

$$= p_k.u^j + q_k.v^j + C_Q^{j+1}.2^{-Q} + C_{2Q}^{j+1}.2^{-2Q} + \ldots\ldots + C_{\left[\frac{P}{Q}-1\right]Q}^{j+1}.2^{-\left[\frac{P}{Q}-1\right]Q} \qquad (9.3.19)$$

$$h_k(u^j,\ v^j,\ C_Q^{j+1},\ C_{2Q}^{j+1},\ \ldots\ldots,\ C_{\left[\frac{P}{Q}-1\right]Q}^{j+1})$$

$$= -q_k.u^j + p_k.v^j + C_Q^{j+1}.2^{-Q} + C_{2Q}^{j+1}.2^{-2Q} + \ldots\ldots + C_{\left[\frac{P}{Q}-1\right]Q}^{j+1}.2^{-\left[\frac{P}{Q}-1\right]Q} \qquad (9.3.20)$$

Similarly, functions g'_k and h'_k are defined in a manner identical to f' in equation (5.2.12) and functions g''_k and h''_k are defined in a manner identical to f" in equation (5.2.13). The values of g'_k, h'_k and g''_k, h''_k are given by equations (5.2.12) and (5.2.13) respectively by substituting Q=B in the equations. In the above equations, the values u^j, v^j = 0 or 1.

The functions g_k and h_k in equations (9.3.19) and (9.3.20) can therefore be implemented with a CBFCS memory each as described in chapter five previously. It should be noted the number of values k takes despends on at what stage of FFT is the butterfly in figure 9.3.1, represented by equations (9.3.2) and (9.3.3), applied to. Thus, the size of the CBFCS memory required in equations (9.3.19) and (9.3.20) varies from stage to stage. The operational speed of this serial-mode implementation depends on the access time of the CBFCS memory and the summation time of the "Adderless-Multiplierless Unit" (AMU) used since the hardware structure is similar to that shown in figure 5.3.3 in chapter five. Finally, to increase the speed of the pipeline FFT processor, one can often use the high-speed parallel-mode implementation of the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) algorithm for realizing the radix-2 FFT butterfly.

CHAPTER TEN

CONCLUSION

So far, in this piece of work, we have presented and investigated a novel design technique and hardware implementation of digital filters for high-speed realtime filtering. Furthermore, a generalized quantization noise model for the analysis of finite-word-length effects on the proposed hardware implementation has also been subsequently presented and investigated. Moreover, the logic design of a demonstration processor built to verify experimentally the proposed hardware implementation and to investigate the above generalized quantization noise model, has been thoroughly described. Additionally, some future work in the improvement of dynamic range and noise performance in the proposed hardware implementation has been suggested along with some applications of the proposed design technique and hardware implementation of digital filters for high-speed realtime filtering.

It is concluded that the proposed design technique is very suitable for synthesizing narrowband filters and filter banks for high-speed realtime digital filtering. In this design technique, the use of a proposed frequency-sampling procedure results in a closed-form expression for a linear-phase equiripple passband structure which is further optimized by a proposed optimization procedure to yield a final optimum transfer function. Thus, the proposed design technique involves two steps. First, we sample, in frequency, the ideal spectrum of a digital filter to be approximated. The sampling is performed by distributing elemental filter responses, called the frequency samples, uniformly across the passband, so that the entire spectrum of interest is sampled "adequately". Next, we erect frequency samples, uniformly in the transition band(s), called the transition samples, at closer equispaced intervals than those in the passband, to improve the transition characteristics and to provide increased flexibility in locating the cutoff frequencies in the passband and the stopband(s). The elemental filters used in the proposed design technique are high-Q resonators. In the second step

of the proposed design technique, we optimize the design to provide an optimum structure. In particular, the transition frequency samples are being optimized to yield the steepest transition skirt(s) possible. Under the proposed optimization procedure, the resultant or final filter response is tailored to suit individual specifications given, and in general, larger stopband attentuation and more reduced sidelobes can be obtained. Linear phase within the passband is maintained by having all poles at equal distance from the unit circle and also "phase-modified". Nevertheless, a novel feature of the above proposed design technique is the use of optimized high-Q resonators as elemental filters to sample, in frequency, along a circle within the unit circle, at two different sets of equispaced sampling points (one in the passband and the other in the transition band(s)), thereby producing a closed-form expression to approximate a desired response. An additional feature of the proposed design technique is the possibility of realtime calculations of the filter coefficients. Nevertheless, the proposed design technique offers a reduction in hardware, even when implemented with existing approaches in hardware implementation. This is feasible via the modularity of the resultant design which allows both pole- and zero-sharing as well as realtime computations of filter coefficients, so as to reduce the storage requirement of, for instance, a filter bank implementation.

In the investigation of the proposed design technique, the filters tested were also quantized, due to the effects of finite word lengths, by a proposed quantized design procedure which is a modified version of the previously mentioned optimization procedure to take into account the effects of finite word lengths. Analyses of the results obtained in the above investigation of the proposed design technique show a satisfactory agreement between theory and practice. In particular, the proposed generalized quantization noise model has been verified accordingly. Furthermore, it is concluded that the output noise of a quantized filter designed by the proposed approach is inversely proportional to the distance of the poles of the elemental filters from the unit circle and also dependent on their resonance angles, when the filter is implemented by the proposed hardware implementation. Consequently, in the suggestions of future work, an

alternative implementation has been proposed, using which, the output noise of a quantized filter designed by the proposed design technique no longer depends on the resonance angles of the elemental filters.

Furtheremore, the experimental results obtained in the above investigation of the proposed design technique, have also verified the proposed hardware implementation and the design of the above demonstration processor, since they agreed with results from exact simulations using the same word length as that of the demonstration processor. The proposed hardware implementation does not include "adds" and "multiplies" of conventional arithmetic units as such, (since no conventional arithmetic units have been used) but instead, employs a novel algorithm, which we have called the "Carry Brought Forward Compensation Scheme" (CBFCS), in order to implement a proposed computational unit, which we have called the "Adderless-Multiplierless Unit" (AMU), to replace conventional arithmetic units, substituting table lookup for computation. Due to the rapidly increasing speed and capacity of semiconductor memory per unit of power dissipation, the proposed hardware implementation capitalizes on memory to offer significant reductions in cost and power consumption for the same speed of operation as that of existing approaches. In fact, the proposed "Carry Brought Forward Compensation Scheme" (CBFCS) makes possible operational speeds which cannot be achieved by existing approaches by "sectioning" the carry propagation path in an addition process, which is the fundamental limitation in speed of all conventional processors.

Having presented and discussed in detail a proposed generalized quantization noise model for the above proposed hardware implementation, the logic design of a finite-wprd-length demonstration processor is also presented in this piece of work. The logic design demonstrates how to put the above proposed "Carry Brought Forward Compensation Scheme" (CBFCS) into practice and the problems thereof. The design is at system level and the resultant architecture is very suitable for large-scale-integration. Moreover, a number of special techniques have been employed in the design of the individual operational units of the demonstration processor. In particular, the proposed "Selective Rounding Scheme" has also been implemented in terms of actual hardware

in the demonstration processor. This scheme has also been experimentally verified.

In deriving the above mentioned generalized quantization noise model for the error analysis of the proposed hardware implementation, the effects of the use of finite word lengths have also been thoroughly explored. The difference in noise performance of the proposed hardware implementation and other existing approaches has also been pointed. Explicit expressions have also been given to assist the designer to access the noise performance of any filter implemented using the proposed approach. It is further concluded that the proposed hardware implementation results in an asymptotically linear filter as the word length used in the implementation increases.

Finally, since the proposed design technique is particularly suitable for designing filter banks, certain applications in the field of digital spectrum analysis, speech synthesis and analysis as well as FM-CW radar signal processing have been suggested. Furthermore, due to the high operational speed of the proposed hardware implementation, its applications extends to the implementation of high-speed Fast Fourier Transformers.

APPENDIX A

THE Z TRANSFORM:

If a continuous-time signal f(t) is sampled into a discrete-time sequence every T seconds, where T is the sampling period, then the sampled waveform f'(t) can be represented as a train of impulses occurring at the sampling instants, 0, T, 2T, 3T,...., the strengths of the individual impulses being equal to the values of the input function f(t) at the respective instants. Thus we can write

$$f'(t) = f(t).\delta_T(t) \tag{A1}$$

where the superscript ' is used to indicate that the function f'(t) is a discrete-time sampled function and $\delta_T(t)$ represents a periodic train of unit impulses spaced T seconds apart

$$\delta_T(t) = \sum_{n=-\infty}^{\infty}\delta(t-nT) \tag{A2}$$

If f(t)=0 for t<0, then equation (A1) becomes

$$f'(t) = \sum_{n=0}^{\infty}f(nT).\delta(t-nT) \tag{A3}$$

Now taking the Laplace Transform of f'(t), we have

$$F'(s) = \mathcal{L}\left[f'(t)\right]$$

$$= \mathcal{L}\left[\sum_{n=0}^{\infty}f(nT).\delta(t-nT)\right]$$

$$= \sum_{n=0}^{\infty}f(nT).e^{-snT} \tag{A4}$$

Since the Laplacian variable s appears in equation (A4) only in the exponential factor, it is convenient to introduce a new symbol $z=e^{sT}$, such that

$$F'(z) = \sum_{n=0}^{\infty}f(nT).z^{-n}$$

$$= Z\left[f'(t)\right] \tag{A5}$$

We have introduced in equation (A5) the symbol z^{-1} which is now being defined as the unit Delay Operator i.e. a delay in time of one sampling period of the system. The previously used symbol z is similarly defined as the unit Advance Operator of the system and is known as the Standard Z Transform variable. In addition, equation (A5) above defines the Z Transform of the sampled analogue waveform f'(t). In the following formal definition of the Z Transform in the discrete-time domain, we shall assume that all signals are discrete-time signals and drop the previously used superscript '.

When a sequence of numbers x_n is identically zero for negative discrete-time (i.e. $x_n=0$ for n=-1, -2, -3, ...) its Z Transform is known as one-sided and defined by

$$X(z) = Z\left[x_n\right]$$

$$= x_o + x_1.z^{-1} + x_2.z^{-2} + ...$$

$$= \sum_{n=0}^{\infty} x_n.z^{-n} \qquad (A6)$$

If x_n is obtained by sampling an analogue signal x(t), it may be designated as x(nT). In essence, the Z Transform is the sampled-data discrete-time version of the Laplace Transform, such that the application of the Z Transform in the solutions of linear difference equations with constant coefficients by transforming the discrete-time variable into the z-plane variable is analogous to solving linear differential equations by the use of the Laplace Transform.

In many cases, sequences are defined over both positive and negative values of n so that, in such cases, a somewhat more general point of veiw is called for to modify equation (A6) as

$$X(z) = \sum_{n=-\infty}^{\infty} x_n.z^{-n} \qquad (A7)$$

It should be noted that the common usage is to call equation (A6) simply the Z Transform and equation (A7) the Two-sided Z Transform, as opposed to the One-sided Z Transform for positive discrete-time. Moreover, the Z Transform is sometimes known as the Standard Z Transform.

It is possible to think of the Z Transform as simply a formal series whose properties can be tabulated, and which never need be summed. However, it is generally preferable to realise that both equations (A6) and (A7), when convergent, are Laurent series in the complex variable z. As such, all the properties of the Laurent series apply. For example, if the series in equation (A6) converges, it must converge in a region $|z| > R_+$. If the series of equation (A7) converges, it must converge in an annular region $R_+ < |z| < R_-$, where R_+ may be zero and R_- may be infinity. The coefficients of a Laurent series are determined by an integral relationship. In the context of the Z Tranzform, this relation is

$$x_n = \frac{1}{2\pi j} \oint_c X(z) . z^{n-1} dz \qquad (A8)$$

where C is a closed contour inside the region of convergence of the power series and enclosing the origin. Equation (A8) is referred to as the Inverse Z Transform.

In the region of convergence of the series, equations (A6) and (A7) represent analytic functions of the complex variable z. These functions can often be extended by analytic continuation everywhere except at certain singular points (poles). Since these singularities of the Z Transform are characteristics of the particular sequence, it is common to plot their locations in the z-plane,(i.e. the complex plane dtermined by the real and imaginary parts of z). It should be noted that it is often convenient, because of the functional form which characterises exponential sequences, to plot singularities in the z^{-1}-plane. Furthermore, some authors define the Z Transform as

$$\tilde{X}(z) = \sum_{n=-\infty}^{\infty} x_n . z^n \qquad (A9)$$

Clearly, equation (A9) is related to our definition by

$$\tilde{X}(z) = X(z^{-1}) \qquad (A10)$$

If either equation (A9) or the z^{-1}-plane is encountered, it is a simple matter to replace z with z^{-1} in order to relate the Z Transform definitions as given in the z- and z^{-1}-planes.

APPENDIX B

ANALYSIS OF A SECOND-ORDER ELEMENTAL FILTER:

Consider the second-order difference equation:

$$y_n = x_n + b_1 \cdot y_{n-1} + b_2 \cdot y_{n-2} \tag{B1}$$

with initial conditions $y_{-1} = y_{-2} = \ldots\ldots = y_{-} = 0$ (B2)

Taking the Z Transform of equation (B1) we have

$$Y(z) = \frac{1}{z^2 - b_1 . z - b_2} . X(z)$$

$$= H(z) . X(z) \tag{B3}$$

where $Y(z)$ is the Z Transform of the output y_n, $X(z)$ is the Z Transform of the input x_n and $H(z)$ is the transfer function of the second-order elemental filter represented by equation (B1).

IMPULSE RESPONSE:

Let the input x_n to the elemental filter be the unit impulse δ_{no}, then $X(z)=1$.

The impulse response h_n of the elemental filter is then obtained via the inverse Z Transform of equation (B3) as follows:

$$h_n = \frac{1}{2\pi j} \oint_c \frac{z^{n+1}}{(z-p_1)(z-p_2)} . dz$$

$$= \frac{1}{p_1 - p_2} (p_1^{n+1} - p_2^{n+1}) \tag{B4}$$

where $$p_{1,2} = \tfrac{1}{2} b_1 \pm \sqrt{\tfrac{1}{4} b_1^2 + b_2} \tag{B5}$$

are the poles of the transfer function $H(z)$. Consider the case when these poles are complex, that is, $\frac{1}{4} b_1^2 + b_2 < 0$. Here we have,

$$p_1 = Re^{j\omega_4 T} \tag{B6}$$

and $$p_2 = Re^{-j\omega_4 T} \tag{B7}$$

where $$R^2 = -b_2 \quad \text{(B8)}$$

and $$2R\cos\omega_r T = b_1 \quad \text{(B9)}$$

Substituting equations (B6)-(B9) into (B4), we have the impulse response of the second-order elemental filter re-written as:

$$h_n = \frac{R^n}{\sin\omega_r T}.\sin(\overline{n+1}\,\omega_r T) \quad \text{(B10)}$$

where ω_r is the resonant frequency of the elemental filter. From equation (B10), it is clear that the above impulse response will be unbounded when $R>1$. This implies an unstable system. Hence, for a stable elemental filter, the value of R is chosen to be less than unity. This is, therefore, in accordance with the system stability criterion which requires that the poles of a stable system should be confined within the unit circle.

FIG: B1 IMPULSE RESPONSE FOR R=1 (MARGINALLY UNSTABLE)

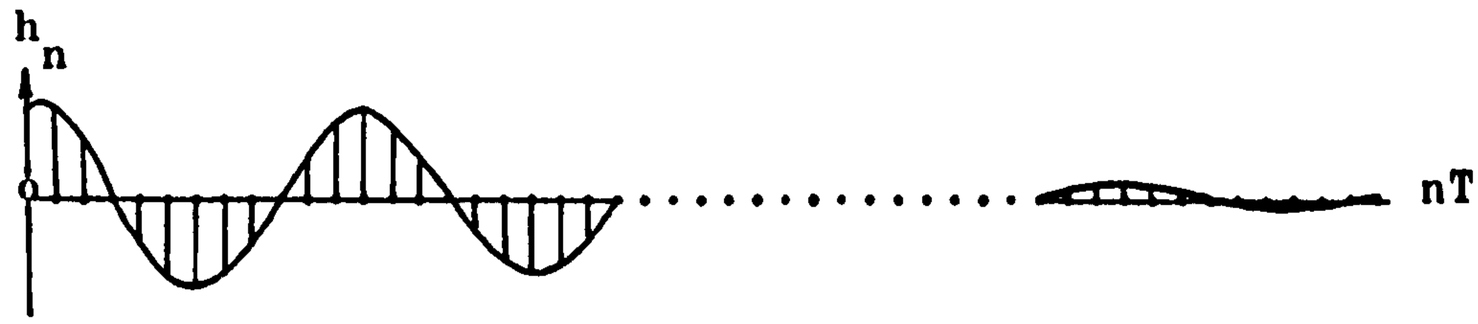


FIG: B2 IMPULSE RESPONSE FOR R≈1 (STABLE)

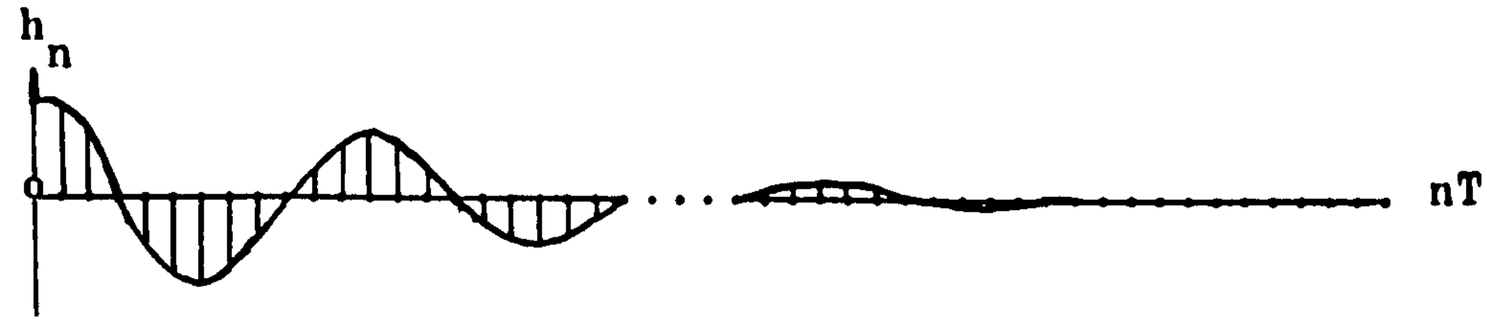


FIG: B3 IMPULSE RESPONSE FOR R<1 (STABLE)

Figure B1 shows a limiting case when the complex poles of an elemental filter, represented by equation (B1), are exactly on the unity circle, that is when R=1. Its impulse is found by equation (B10) to be a constant amplitude sinusoid oscillating at a natural frequency ω_r. Although its impulse response is bounded, such an elemental filter is on the verge of instability, and is therefore discarded in the choice of elemental filters for the proposed design technique discussed in chapter three.

On the other hand, when the complex poles are located within the unit circle, that is when R<1, then the elemental filter is stable. Figure B3 depicts the impulse response of such an elemental filter, which is found by equation (B10) to represent a damped sinusoid decaying exponentially to zero, but meanwhile oscillating at a natural frequency ω_r. The duration of the impulse response is further seen to be proportional to the value of R. Hence, when the value of R approaches unity, then the duration of the above impulse response becomes longer and longer. This is in accordance with the fact that the Q-factor of the elemental filter increases when the value of R approaches unity. Figure B2 shows the impulse response of a high-Q elemental filter whose complex poles are located slightly within the unit circle, that is when R≈1. Such an elemental filter is made use of in the proposed design technique discussed in chapter three. The choice of such high-Q elemental filters in the above proposed design technique will be made clear in the next section where the frequency and phase responses of an elemental filter will be discussed.

FREQUENCY AND PHASE RESPONSES:

When the input to an elemental filter represented by equation (B1) is

$$x_n = \sin(n\omega T) \qquad \text{(B11)}$$

and its complex poles are within the unit circle (that is R<1 and the filter is stable), then its steady-state output is

$$y_n = \left|H(e^{j\omega T})\right| \sin(n\omega T + \angle H(e^{j\omega T})) \qquad \text{(B12)}$$

where $\left|H(e^{j\omega T})\right|$ and $\angle H(e^{j\omega T})$ are the frequency and phase responses of

the elemental filter respectively given by

$$\left|H(e^{j\omega T})\right| = \left[\frac{1}{(1-b_1\cos\omega T-b_2\cos 2\omega T)^2 + (b_1\sin\omega T+b_2\sin 2\omega T)^2}\right]^{\frac{1}{2}} \quad \text{(B13)}$$

$$\text{and } \angle H(e^{j\omega T}) = -\tan^{-1}\left[\frac{b_1\sin\omega T+b_2\sin 2\omega T}{1-b_1\cos\omega T-b_2\cos 2\omega T}\right] \quad \text{(B14)}$$

Figure B4 and figure B5 below show the frequency and phase responses given by the above equations for varying R and $\omega_r = \pi/4T$.

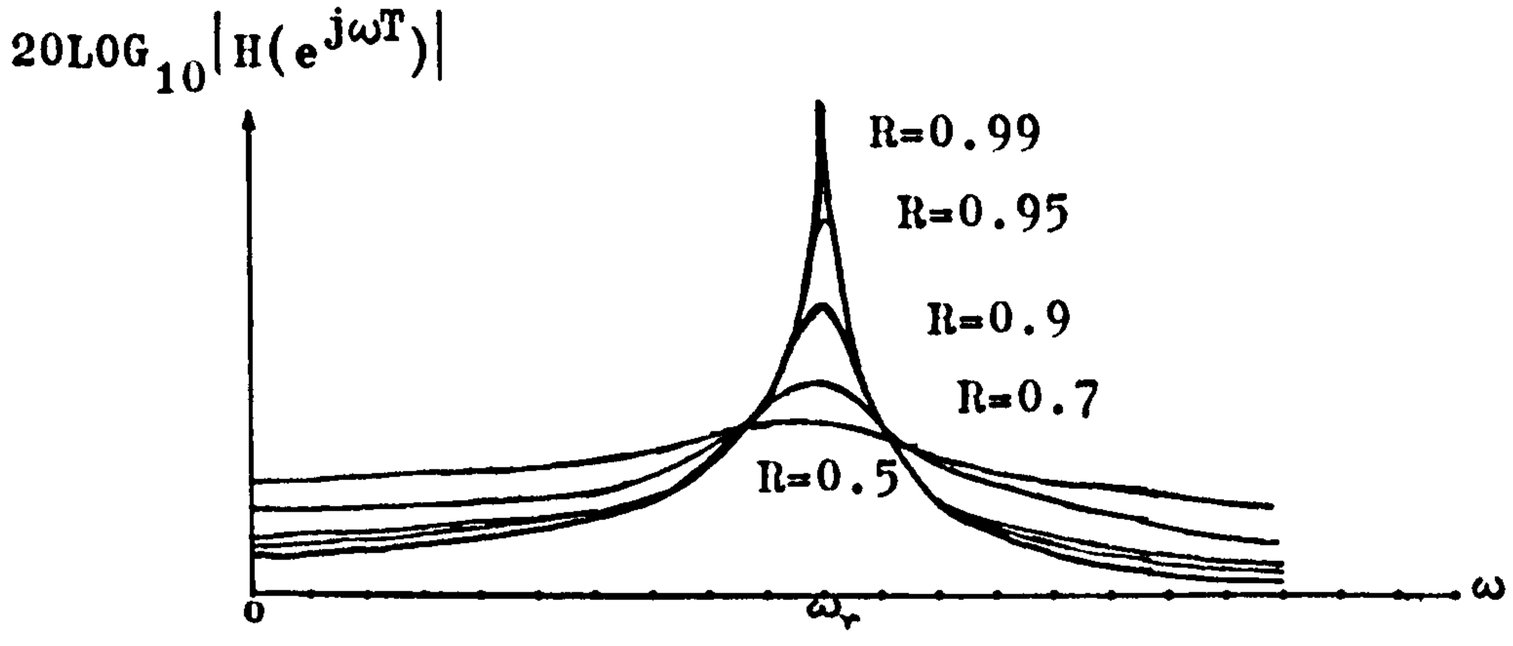


FIG: B4 FREQUENCY RESPONSES FOR VARYING R ($\omega_r = \pi/4T$)

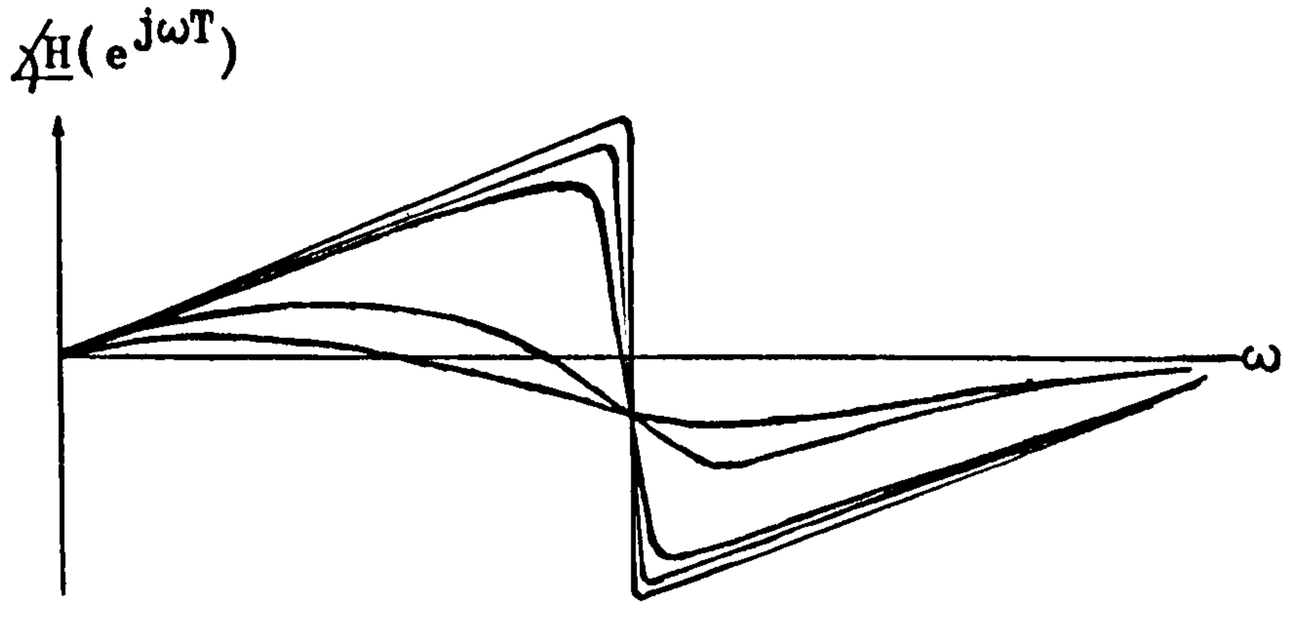


FIG: B5 PHASE RESPONSES FOR VARYING R ($\omega_r = \pi/4T$)

From the above plots, it is clear that the second-order elemental filters are simple digital resonators. It is further seen that the gain of an elemental filter is proportional to the value of R. Hence, low-Q resonators have low gains and non-linear phase responses which are not suitable for the proposed design technique

discussed in chapter three. Moreover, frequency aliasing effects are obvious for low-Q resonators. However, with high-Q resonators high gains are obtained and their phase responses become more and more linear. In the limit, when R approaches unity, we have

$$\lim_{R\to 1} \angle H(e^{j\omega T}) = -\tan^{-1}\left\{\frac{2\cos\omega_r T\sin\omega T - 2\sin\omega T\cos\omega T}{1 + \cos 2\omega T - 2\cos\omega_r T\cos\omega T}\right\}$$

$$= -\tan^{-1}(\tan(-\omega T)) \qquad \text{(B15)}$$

and since the tangent of an angle is negative in the 2nd and 4th quadrants, we have

$$\lim_{R\to 1} \angle H(e^{j\omega T}) = \begin{cases} \omega T & \text{for } 0<\omega<\omega_r \\ (\omega T-\pi) & \text{for } \omega_r<\omega<(\frac{2\pi}{T}-\omega_r) \\ (\omega T-2\pi) & \text{for } (\frac{2\pi}{T}-\omega_r)<\omega<\frac{2\pi}{T} \end{cases} \qquad \text{(B16)}$$

Hence, $$\lim_{R\to 1} \frac{d}{d\omega} \angle H(e^{j\omega T}) = T \qquad \text{(B17)}$$

which is a constant. Consequently, linear phase can be obtained with high-Q resonators whose values of R approach unity as above (that is the poles of these high-Q resonators are very close to the unit circle). This is essential in the approximation of linear phase in the proposed design technique discussed in chapter three. Hence, high-Q resonators are used as elemental filters in the above proposed design technique for their high gains and phase responses. In addition to the above mentioned, high-Q resonators have very narrowband frequency responses where frequency aliasing effects are minimized when the Impulse Invariance Transformation is used in their derivations which are discussed next.

DERIVATION OF ELEMENTAL FILTER EXPRESSIONS:

Digital resonators can be derived from analogue tuned-circuits in the s-plane via the Impulse Invariance Transformation. Consider a pole-residue function in the s-plane as follows:

$$H(s) = \sum_{i=1}^{N} \frac{B_i}{s+\alpha_i} \tag{B18}$$

where the residue $$B_i = H(s).(s+\alpha_i)\Big|_{s=-\alpha_i} \tag{B19}$$

The impulse response of the above function is:

$$h(t) = \sum_{i=1}^{N} B_i.e^{-\alpha_i.t}.u_{-1}(t) \tag{B20}$$

The corresponding digital impulse response is:

$$h_n = \sum_{i=1}^{N} B_i.e^{-\alpha_i nT}.u_{-n} \tag{B21}$$

Taking the Z Transform of h_n we have

$$H(z) = \sum_{n=o}^{\infty} h_n.z^{-n}$$

$$= \sum_{n=o}^{\infty} \sum_{i=1}^{N} B_i.e^{-\alpha_i nT}.z^{-n} \tag{B22}$$

Interchanging orders of summation in the above equation, we have

$$H(z) = \sum_{i=1}^{N} B_i \sum_{n=o}^{\infty} (e^{-\alpha_i T}.z^{-1})^n$$

$$= \sum_{i=1}^{N} \frac{B_i}{1 - e^{-\alpha_i T}.z^{-1}} \tag{B23}$$

Hence, we have the first-order transform:

$$\frac{B_i}{s+\alpha_i} \longrightarrow \frac{B_i}{1-z^{-1}.e^{-\alpha_i T}} \tag{B24}$$

For complex poles, when α_i is complex, the residue B_i is also complex. However, $h(t)$ is real which implies that there is also a conjugate pole at α_i^* with residue B_i^*. Grouping terms, we have

$$\frac{B_i}{s+\alpha_i} + \frac{B_i^*}{s+\alpha_i^*} = \frac{(B_i+B_i^*)s + B_i.\alpha_i^* + B_i^*.\alpha_i}{s^2 + (\alpha_i + \alpha_i^*)s + \alpha_i.\alpha_i^*} \tag{B25}$$

If we let $\alpha_i = x_i + jy_i$ and $B_i = u_i + jv_i$ and applying equation (B24) to (B25) then

$$\frac{B_i}{1-z^{-1}.e^{-\alpha_i T}} + \frac{B_i^*}{1-z^{-1}.e^{-\alpha_i^* T}}$$

$$= \frac{2u_i - z^{-1}.e^{-x_i T}(2u_i\cos(y_i T) - 2v_i\sin(y_i T))}{1-z^{-1}.2e^{-x_i T}\cos(y_i T) + z^{-2}.e^{-2x_i T}} \qquad \text{(B26)}$$

Comparing equation (B26) with equation (B25) we have,

$$\frac{s+x}{s^2+2xs+(x^2+y^2)} \longrightarrow \frac{1-z^{-1}.e^{-xT}\cos(yT)}{1-z^{-1}.2e^{-xT}\cos(yT)+z^{-2}.e^{-2xT}} \qquad \text{(B27)}$$

$$\frac{y}{s^2+2xs+(x^2+y^2)} \longrightarrow \frac{z^{-1}.e^{-xT}\sin(yT)}{1-z^{-1}.2e^{-xT}\cos(yT)+z^{-2}.e^{-2xT}} \qquad \text{(B28)}$$

which are the required second-order digital resonators. We shall next consider the gains of different types of second-order digital resonators below.

GAINS OF ELEMENTAL FILTERS:

A digital resonator, in practice, can be specified by placing the pair of complex conjugate poles, and in most cases, a single zero in the z-plane. Hence, we shall consider the general case of a second-order resonator with poles at $z=Re^{\pm j\omega_r T}$ and a zero at P. The transfer function of this general second-order resonator is therefore written as:

$$H(z) = \frac{1-z^{-1}.P}{1-z^{-1}.2R\cos(\omega_r T)+z^{-2}.R^2} \qquad \text{(B29)}$$

By the cosine rule we have,

$$\left|H(e^{j\omega T})\right| = \left\{\frac{1+P^2-2P\cos\omega T}{(1+R^2-2R\cos(\omega-\omega_r)T)(1+R^2-2R\cos(\omega+\omega_r)T)}\right\}^{\frac{1}{2}} \qquad \text{(B30)}$$

Thus the gain of the general second-order resonator depends on the choice of zeros, in addition to the value of R.

When $P=\cos\omega_r T$ and for values of R close to unity, the gain of the general second-order resonator at the resonance frequency can be approximated by:

$$\left| H(e^{j\omega T}) \right| = \frac{1}{2(1-R).\sqrt{R}} \qquad (B31)$$

which is independent of ω_r. Thus this choice of P makes possible the design of an equal-gain bank of resonators covering a wide range of frequencies. However, for narrowband resonators, that is R≈1, equation (B31) shows that the gain at resonance is usually appreciably greater than unity. Hence, in hardware implementations, knowledge of elemental filter gains is required for the determination of appropriate word lengths and to avoid overflow problems.

By making $P=R\cos\omega_r T$, the difference equation for H(z) can be written as:

$$y_n = R\cos\omega_r T(2y_{n-1} - x_{n-1}) - R^2.y_{n-2} + x_n \qquad (B32)$$

which only requires two multiplications as opposed to four which are required in the general case. Hence, this choice of P saves on computational efforts and is very suitable for realtime digital filtering.

Finally, when P=1, that is zero gain at $\omega=0$, which is often desirable, then two multiplications are required as in the above case. This choice of P still preserves the frequency response at resonance and is also very suitable for realtime digital filtering. In fact, in the proposed design technique discussed in chapter three, this choice of P makes possible the zero-sharing property which also saves on hardware in the physical implementations of elemental filters.

Having discussed the gain of a general second-order resonator and the choice of zeros for elemental filters, we shall next discuss the use of high-order elemental filters which can also be used in the above design technique proposed in chapter three.

HIGHER-ORDER ELEMENTAL FILTERS:

In the proposed design technique in

chapter three, high-Q second-order elemental filters with sufficient gain were used. In theory, it is possible to replace a high-Q second-order resonator by a cascade of n low-Q second-order resonators centred at the same resonance frequency ω_r. Consider a higher-order elemental filter made up of n normalized second-order resonators represented as

$$H_n(z) = \left[\frac{(1-R)\sqrt{1-2R\cos 2\omega_r T+R^2}\ z^2}{z^2-z.2R\cos\omega_r T+R^2}\right]^n \qquad (B33)$$

where the factor $(1-R)\sqrt{1-2R\cos 2\omega_r T+R^2}$ is the reciprocal of the gain at resonance of each second-order resonator in cascade, such that the gain at resonance of each normalized second-order resonator above is now unity. (Normalization of filter gain is often required to prevent overflow problems). Figure B6 shows a plot of the gain of $H_n(z)$ versus frequency for the cases of n=1 and n=4.

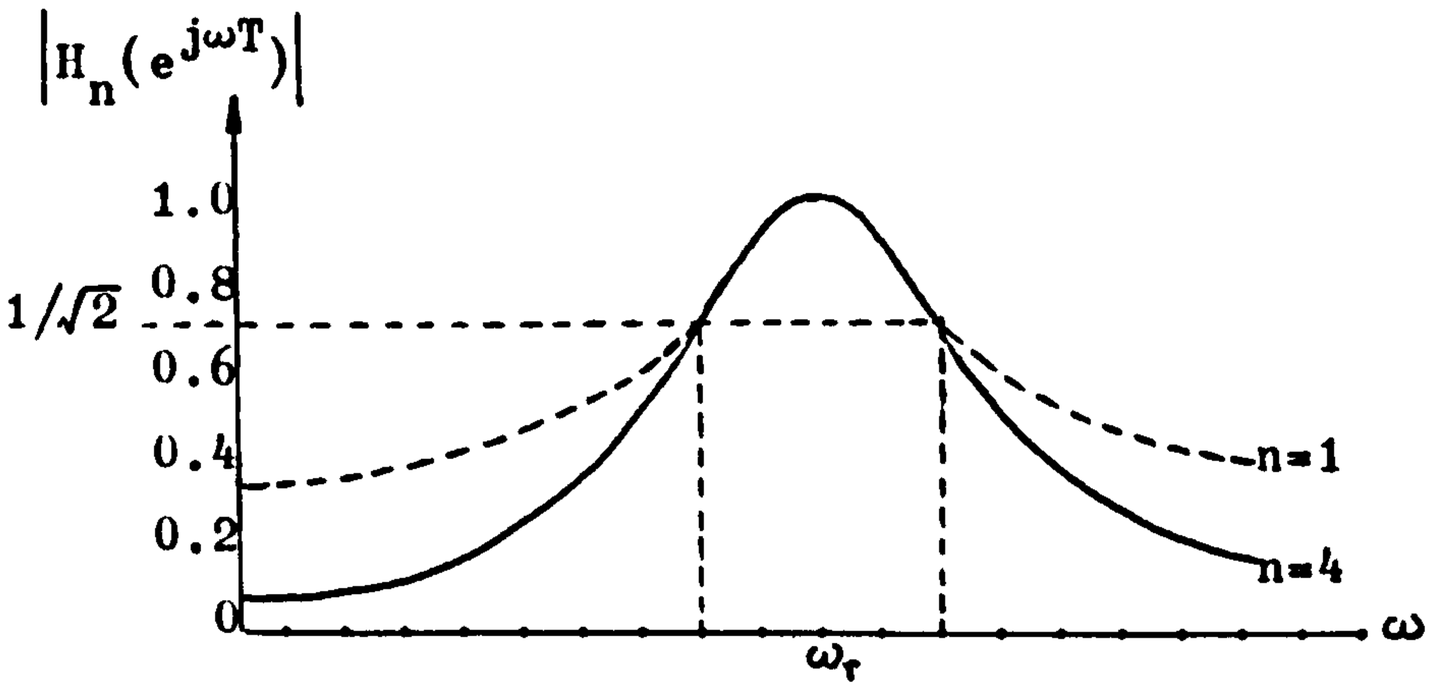


FIG: B6 FREQUENCY RESPONSE OF A HIGHER-ORDER ELEMENTAL FILTER IN COMPARISON WITH THAT OF A SECOND-ORDER ELEMENTAL FILTER

From the figure above, it is seen that the frequency response of the elemental filter for n=4 is much sharper than that for n=1. Furthermore, an extra 14dB of stopband attentuation is achieved in the above example where a high-Q higher-order elemental filter is formed by cascading four low-Q (R≈0.7) second-order resonators. These four low-Q second-order resonators used are seen to have the

same 3dB bandwidth as the high-Q resultant elemental filter. In practice, scaling constants are required in between stages in the above cascade to avoid overflow problems.

In conclusion, it is found that a tradeoff exists between the choice of high-Q second-order elemental filters and the use of a higher-order elemental filter formed by a cascade of n identical low-Q second-order resonators. When high-Q second-order elemental filters are used in the proposed design technique discussed in chapter three, the resultant design has then a lower order than that where higher-order elemental filters are used. Hence, the above mentioned tradeoff is in terms of filter order of the resultant design, when finite word length effects are ignored.

APPENDIX C

In this appendix, we include the contents of the CBFCS memories of the second-order elemental filters pertaining to the eighth-order bandpass filters in chapter eight. As explained in section 5.3.1 in chapter five, each of the above CBFCS memories can further be divided into the original memory and the extra memory whose individual contents are calculated according to equation (5.2.10) as described in section 5.2 in chapter five.

Since the eighth-order bandpass filters in chapter eight are realized in the parallel form, the maximum number of locations in each CBFCS memory (tables C1 to C17) is 32 (c.f. equation (5.3.1) for a general second-order difference equation). Hence, in tables C1 to C17, the first 16 locations out of the total of 32 in each table are for the original memory when the brought forward carry $C_{8,n}^{j+1}=0$ while the other $\varkappa_{max}=16$ are for the extra memory when the above brought forward carry $C_{8,n}^{j+1}=1$.

However, as explained in section 5.3.1, the actual number of locations in each extra memory can be reduced to $\varkappa$ where $\varkappa<\varkappa_{max}=16$. Furthermore, each of these $\varkappa$ locations is of eight bits long instead of 16 bits long, as in the original memory, so that the actual size of each extra memory is only $8\varkappa$ bits. For instance, the $\varkappa_{max}=16$ locations shown in table C1 can be reduced to $\varkappa=10$ locations, each of eight bits long, giving the size of the extra memory to be 80 bits, that is, a reduction of 176 bits in memory requirement.

By the same token, the locations of the extra memories shown in tables C2 to C17 can actually be reduced to 8, 8, 10, 10, 8, 8, 8, 3, 8, 8, 7, 4, 2, 0, 5 and 6 locations respectively. Thus, by following the method previously outlined in section 5.3.1, one can then achieve a total memory size reduction of (176 + 192 + 192 + 176) = 736 bits for the eighth-order bandpass filter corresponding to tables C1 to C4. Also, a total memory size reduction of (176 + 192 + 192 + 192) = 752 bits for the eighth-order bandpass filter corresponding to tables C5 to C8 and, for the eighth-order bandpass filter coresponding to tables C9 to C12, a total memory size reduction of (232 + 192 + 192 + 200) = 816 bits. For the eighth-order bandpass filter corresponding

to tables C13 to C16, the reduction in memory size is 936 bits, while for the second-order elemental filter corresponding to table C17, the reduction is 208 bits.

In the conclusion of chapter five, a comparison was made between the proposed CBFCS approach and B.Liu et.al. approach. In this comparison, the size of the extra memory required for the implementation of a general second-order difference equation and its cost were calculated on the basis of using $\chi=\chi_{max}$ which, therefore, represented the maximum extra memory size and cost respectively. As seen in the above examples, $\chi\neq\chi_{max}$ in general which further makes our comparison more attractive. In fact, the size of the extra memory also depends on the form of realization. In the general second-order difference equation discussed in chapter five, the cascade form of realization requires more memory bits than the parallel form since the argument of the function f in equation (5.2.10) has one more variable for the cascade form than for the parallel form. In the proposed design technique in chapter three, we have chosen the parallel form of implementation which further makes the hardware implementations of filters designed by the above proposed design technique more cost-effective.

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	y_{n-1}^j	y_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0		1	1	1	0	0	0	0	0	1	0	1	0	1	0	0
0	0	1	0	0		0	0	1	1	0	1	0	0	0	0	0	0	1	1	1	1
0	0	1	1	0	16	0	0	0	1	0	1	0	0	1	1	1	0	0	0	1	1
0	1	0	0	0	L	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	0
0	1	0	1	0	0	1	1	0	0	0	1	1	0	1	1	0	0	1	1	0	0
0	1	1	0	0	C	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	1
0	1	1	1	0	A	1	1	1	1	1	0	1	0	1	1	0	1	1	0	1	1
1	0	0	0	0	T	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	I	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0
1	0	1	0	0	O	0	1	0	1	0	1	0	0	0	0	0	0	1	1	1	1
1	0	1	1	0	N	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	1
1	1	0	0	0	S	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0
1	1	0	1	0	↓	1	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0
1	1	1	0	0		0	0	1	1	1	0	1	0	0	0	0	0	0	1	1	1
1	1	1	1	0		0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	1
0	0	0	0	1	↑	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1		1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	0
0	0	1	0	1		0	0	1	1	0	1	0	0	1	0	0	0	1	1	1	1
0	0	1	1	1	x_{max}	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1
0	1	0	0	1	L	1	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0
0	1	0	1	1	O	1	1	0	0	0	1	1	1	0	1	0	0	1	1	0	0
0	1	1	0	1	C	0	0	0	1	1	0	1	0	1	0	0	0	0	1	1	1
0	1	1	1	1	A	1	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1
1	0	0	0	1	T	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	I	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0
1	0	1	0	1	O	0	1	0	1	0	1	0	0	1	0	0	0	1	1	1	1
1	0	1	1	1	N	0	0	1	1	0	1	0	1	0	1	1	0	0	0	1	1
1	1	0	0	1	S	0	0	0	0	0	1	1	0	0	1	1	1	1	1	0	0
1	1	0	1	1	↓	1	1	1	0	0	1	1	1	0	1	0	0	1	1	0	0
1	1	1	0	1		0	0	1	1	1	0	1	0	1	0	0	0	0	1	1	1
1	1	1	1	1		0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	1

TABLE C1: CBFCS MEMORY FOR THE TRANSITION SAMPLE VALUE W_n = 0.5 IN CONFIGURATION (b) SHOWN IN FIGURE 8.1.2 FOR THE REALIZATION OF THE LOWER TRANSITION SECOND-ORDER ELEMENTAL FILTER $H_1^T(z)$ IN THE 120Hz EIGHTH-ORDER BANDPASS FILTER IN CHAPTER EIGHT.

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	w_{n-1}^j	w_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0		1	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0
0	0	1	0	0		0	0	1	1	0	0	1	1	1	0	0	1	0	1	1	0
0	0	1	1	0	16	0	0	0	1	0	1	0	0	0	1	1	1	1	0	1	0
0	1	0	0	0	L	1	1	1	0	0	1	1	0	0	0	1	1	0	1	0	1
0	1	0	1	0	0	1	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1
0	1	1	0	0	C	0	0	0	1	1	0	0	1	1	1	0	0	1	0	1	1
0	1	1	1	0	A	1	1	1	1	1	0	1	0	1	0	0	1	1	1	1	1
1	0	0	0	0	T	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	I	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0
1	0	1	0	0	O	0	1	0	1	0	0	1	1	1	0	0	1	0	1	1	0
1	0	1	1	0	N	0	0	1	1	0	1	0	0	0	1	1	1	1	0	1	0
1	1	0	0	0	S	0	0	0	0	0	1	1	0	0	0	1	1	0	1	0	1
1	1	0	1	0	|	1	1	1	0	0	1	1	1	0	0	0	0	1	0	0	1
1	1	1	0	0	|	0	0	1	1	1	0	0	1	1	1	0	0	1	0	1	1
1	1	1	1	0	↓	0	0	0	1	1	0	1	0	1	0	0	1	1	1	1	1
0	0	0	0	1	↑	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1	|	1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	0
0	0	1	0	1	|	0	0	1	1	0	1	0	0	0	0	0	1	0	1	1	0
0	0	1	1	1	χ_{max}	0	0	0	1	0	1	0	0	1	1	1	1	1	0	1	0
0	1	0	0	1	L	1	1	1	0	0	1	1	0	1	0	1	1	0	1	0	1
0	1	0	1	1	0	1	1	0	0	0	1	1	1	1	0	0	0	1	0	0	1
0	1	1	0	1	C	0	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1
0	1	1	1	1	A	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1
1	0	0	0	1	T	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	I	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0
1	0	1	0	1	O	0	1	0	1	0	1	0	0	0	0	0	1	0	1	1	0
1	0	1	1	1	N	0	0	1	1	0	1	0	0	1	1	1	1	1	0	1	0
1	1	0	0	1	S	0	0	0	0	0	1	1	0	1	0	1	1	0	1	0	1
1	1	0	1	1	|	1	1	1	0	0	1	1	1	1	0	0	0	1	0	0	1
1	1	1	0	1	|	0	0	1	1	1	0	1	0	0	1	0	0	1	0	1	1
1	1	1	1	1	↓	0	0	0	1	1	0	1	1	0	0	0	1	1	1	1	1

TABLE C2: CBFCS MEMORY FOR THE FIRST PASSBAND ELEMENTAL FILTER -$H_1^P(z)$ IN CONFIGUIATION (b) SHOWN IN FIGURE 8.1.2 OF THE 120Hz EIGHTH-ORDER BANDPASS FILTER IN CHAPTER EIGHT.

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	v_{n-1}^j	v_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 0 1 0	1 0 0 1 1 1 0 1
0	0	1	1	0	16	0 0 0 1 0 0 1 1	0 1 1 1 0 0 0 1
0	1	0	0	0	L	1 1 1 0 0 1 1 0	1 0 1 1 0 0 1 0
0	1	0	1	0	O	1 1 0 0 0 1 1 1	1 0 0 0 0 1 1 0
0	1	1	0	0	C	0 0 0 1 1 0 0 1	0 1 0 0 1 1 1 1
0	1	1	1	0	A	1 1 1 1 1 0 1 0	0 0 1 0 0 0 1 1
1	0	0	0	0	T	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0
1	0	0	1	0	I	0 0 0 0 0 0 0 0	1 1 0 1 0 1 0 0
1	0	1	0	0	O	0 1 0 1 0 0 1 0	1 0 0 1 1 1 0 1
1	0	1	1	0	N	0 0 1 1 0 0 1 1	0 1 1 1 0 0 0 1
1	1	0	0	0	S	0 0 0 0 0 1 1 0	1 0 1 1 0 0 1 0
1	1	0	1	0		1 1 1 0 0 1 1 1	1 0 0 0 0 1 1 0
1	1	1	0	0		0 0 1 1 1 0 0 1	0 1 0 0 1 1 1 1
1	1	1	1	0	↓	0 0 0 1 1 0 1 0	0 0 1 0 0 0 1 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1		0 0 1 1 0 0 1 1	0 0 0 1 1 1 0 1
0	0	1	1	1	x_{max}	0 0 0 1 0 0 1 1	1 1 1 1 0 0 0 1
0	1	0	0	1	L	1 1 1 0 0 1 1 1	0 0 1 1 0 0 1 0
0	1	0	1	1	O	1 1 0 0 1 0 0 0	0 0 0 0 0 1 1 0
0	1	1	0	1	C	0 0 0 1 1 0 0 1	1 1 0 0 1 1 1 1
0	1	1	1	1	A	1 1 1 1 1 0 1 0	1 0 1 0 0 0 1 1
1	0	0	0	1	T	0 0 1 0 0 0 0 0	1 0 0 0 0 0 0 0
1	0	0	1	1	I	0 0 0 0 0 0 0 1	0 1 0 1 0 1 0 0
1	0	1	0	1	O	0 1 0 1 0 0 1 1	0 0 0 1 1 1 0 1
1	0	1	1	1	N	0 0 1 1 0 0 1 1	1 1 1 1 0 0 0 1
1	1	0	0	1	S	0 0 0 0 0 1 1 1	0 0 1 1 0 0 1 0
1	1	0	1	1		1 1 1 0 1 0 0 0	0 0 0 0 0 1 1 0
1	1	1	0	1		0 0 1 1 1 0 0 1	1 1 0 0 1 1 1 1
1	1	1	1	1	↓	0 0 0 1 1 0 1 0	1 0 1 0 0 0 1 1

TABLE C3: CBFCS MEMORY FOR THE SECOND PASSBAND ELEMENTAL FILTER $H_2^p(z)$ IN CONFIGURATION (b) SHOWN IN FIGURE 8.1.2 OF THE 120Hz EIGHTH-ORDER BANDPASS FILTER IN CHAPTER EIGHT.

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	u_{n-1}^j	u_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0		1	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0
0	0	1	0	0		0	0	1	1	0	0	1	0	0	0	0	1	1	1	0	1
0	0	1	1	0	16	0	0	0	1	0	0	1	0	1	1	1	1	0	0	0	1
0	1	0	0	0	L	1	1	1	0	0	1	1	0	1	1	1	1	0	0	0	1
0	1	0	1	0	0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	0	1
0	1	1	0	0	C	0	0	0	1	1	0	0	1	0	0	0	0	1	1	1	0
0	1	1	1	0	A	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1	0
1	0	0	0	0	T	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	I	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0
1	0	1	0	0	O	0	1	0	1	0	0	1	0	0	0	0	1	1	1	0	1
1	0	1	1	0	N	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0	1
1	1	0	0	0	S	0	0	0	0	0	1	1	0	1	1	1	1	0	0	0	1
1	1	0	1	0		1	1	1	0	0	1	1	1	1	1	0	0	0	1	0	1
1	1	1	0	0		0	0	1	1	1	0	0	1	0	0	0	0	1	1	1	0
1	1	1	1	0	↓	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1	0
0	0	0	0	1	↑	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1		1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	0
0	0	1	0	1		0	0	1	1	0	0	1	0	1	0	0	1	1	1	0	1
0	0	1	1	1	x_{max}	0	0	0	1	0	0	1	1	0	1	1	1	0	0	0	1
0	1	0	0	1	L	1	1	1	0	0	1	1	1	0	1	1	1	0	0	0	1
0	1	0	1	1	0	1	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1
0	1	1	0	1	C	0	0	0	1	1	0	0	1	1	0	0	0	1	1	1	0
0	1	1	1	1	A	1	1	1	1	1	0	1	0	0	1	1	0	0	0	1	0
1	0	0	0	1	T	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	I	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0
1	0	1	0	1	O	0	1	0	1	0	0	1	0	1	0	0	1	1	1	0	1
1	0	1	1	1	N	0	0	1	1	0	0	1	1	0	1	1	1	0	0	0	1
1	1	0	0	1	S	0	0	0	0	0	1	1	1	0	1	1	1	0	0	0	1
1	1	0	1	1		1	1	1	0	1	0	0	0	0	1	0	0	0	1	0	1
1	1	1	0	1		0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0
1	1	1	1	1	↓	0	0	0	1	1	0	1	0	0	1	1	0	0	0	1	0

TABLE C4: CBFCS MEMORY FOR THE REALIZATION OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER-$H_u^T(z)$ IN THE 120Hz EIGHTH-ORDER BANDPASS FILTER WITH TRANSITION SAMPLE VALUE W_n=0.5 IN CONFIGURATION (b) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	y_{n-1}^j	y_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0		1	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0
0	0	1	0	0		0	0	1	1	0	1	0	0	0	0	0	0	1	1	1	1
0	0	1	1	0	16	0	0	0	1	0	1	0	0	1	1	1	0	0	0	1	1
0	1	0	0	0	L	1	1	1	1	0	0	1	0	1	1	1	1	1	1	0	0
0	1	0	1	0	0	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	0
0	1	1	0	0	C	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	1
0	1	1	1	0	A	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	0	T	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	I	1	1	1	1	0	0	0	0	1	1	0	1	0	1	0	0
1	0	1	0	0	O	0	1	0	0	0	1	0	0	0	0	0	0	1	1	1	1
1	0	1	1	0	N	0	0	1	0	0	1	0	0	1	1	1	0	0	0	1	1
1	1	0	0	0	S	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0
1	1	0	1	0		1	1	1	0	0	0	1	1	1	1	0	1	0	0	0	0
1	1	1	0	0		0	0	1	1	0	1	1	1	0	0	0	0	1	0	1	1
1	1	1	1	0	↓	0	0	0	1	0	1	1	1	1	1	0	1	1	1	1	1
0	0	0	0	1	↑	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1		1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	0
0	0	1	0	1		0	0	1	1	0	1	0	0	1	0	0	0	1	1	1	1
0	0	1	1	1	χ_{max}	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1
0	1	0	0	1	L	1	1	1	1	0	0	1	1	0	1	1	1	1	1	0	0
0	1	0	1	1	0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	0
0	1	1	0	1	C	0	0	1	0	0	1	1	1	1	0	0	0	1	0	1	1
0	1	1	1	1	A	0	0	0	0	1	0	0	0	0	1	0	1	1	1	1	1
1	0	0	0	1	T	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	I	1	1	1	1	0	0	0	1	0	1	0	1	0	1	0	0
1	0	1	0	1	O	0	1	0	0	0	1	0	0	1	0	0	0	1	1	1	1
1	0	1	1	1	N	0	0	1	0	0	1	0	1	0	1	1	0	0	0	1	1
1	1	0	0	1	S	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0
1	1	0	1	1		1	1	1	0	0	1	0	0	0	1	0	1	0	0	0	0
1	1	1	0	1		0	0	1	1	0	1	1	1	1	0	0	0	1	0	1	1
1	1	1	1	1	↓	0	0	0	1	1	0	0	0	0	1	0	1	1	1	1	1

TABLE C5: CBFCS MEMORY FOR THE REALIZATION OF THE LOWER TRANSITION SECOND-ORDER ELEMENTAL FILTER $H_1^T(z)$ WITH TRANSITION SAMPLE VALUE $W_n = 0.5$ IN CONFIGURATION (a) FOR THE EIGHTH-ORDER BANDPASS FILTER (120Hz) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	w_{n-1}^j	w_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 0 1 1	1 0 0 1 0 1 1 0
0	0	1	1	0	16	0 0 0 1 0 1 0 0	0 1 1 1 1 0 1 0
0	1	0	0	0	L	0 0 0 1 1 0 0 1	1 1 0 0 1 0 1 1
0	1	0	1	0	0	1 1 1 1 1 0 1 0	1 0 0 1 1 1 1 1
0	1	1	0	0	C	0 1 0 0 1 1 0 1	0 1 1 0 0 0 0 1
0	1	1	1	0	A	0 0 1 0 1 1 1 0	0 0 1 1 0 1 0 1
1	0	0	0	0	T	1 1 1 0 0 0 0 0	0 0 0 0 0 0 0 0
1	0	0	1	0	I	1 1 0 0 0 0 0 0	1 1 0 1 0 1 0 0
1	0	1	0	0	O	0 0 0 1 0 0 1 1	1 0 0 1 0 1 1 0
1	0	1	1	0	N	1 1 1 1 0 1 0 0	0 1 1 1 1 0 1 0
1	1	0	0	0	S	1 1 1 1 1 0 0 1	1 1 0 0 1 0 1 1
1	1	0	1	0	|	1 1 0 1 1 0 1 0	1 0 0 1 1 1 1 1
1	1	1	0	0	|	0 0 1 0 1 1 0 1	0 1 1 0 0 0 0 1
1	1	1	1	0	↓	0 0 0 0 1 1 1 0	0 0 1 1 0 1 0 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1		0 0 1 1 0 1 0 0	0 0 0 1 0 1 1 0
0	0	1	1	1	χ_{max}	0 0 0 1 0 1 0 0	1 1 1 1 1 0 1 0
0	1	0	0	1	L	0 0 0 1 1 0 1 0	0 1 0 0 1 0 1 1
0	1	0	1	1	0	1 1 1 1 1 0 1 1	0 0 0 1 1 1 1 1
0	1	1	0	1	C	0 1 0 0 1 1 0 1	1 1 1 0 0 0 0 1
0	1	1	1	1	A	0 0 1 0 1 1 1 0	1 0 1 1 0 1 0 1
1	0	0	0	1	T	1 1 1 0 0 0 0 0	1 0 0 0 0 0 0 0
1	0	0	1	1	I	1 1 0 0 0 0 0 1	0 1 0 1 0 1 0 0
1	0	1	0	1	O	0 0 0 1 0 1 0 0	0 0 0 1 0 1 1 0
1	0	1	1	1	N	1 1 1 1 0 1 0 0	1 1 1 1 1 0 1 0
1	1	0	0	1	S	1 1 1 1 1 0 1 0	0 1 0 0 1 0 1 1
1	1	0	1	1	|	1 1 0 1 1 0 1 1	0 0 0 1 1 1 1 1
1	1	1	0	1	|	0 0 1 0 1 1 0 1	1 1 1 0 0 0 0 1
1	1	1	1	1	↓	0 0 0 0 1 1 1 0	1 0 1 1 0 1 0 1

TABLE C6: CBFCS MEMORY FOR THE REALIZATION OF THE FIRST PASSBAND ELEMENTAL FILTER $-H_1^P(z)$ OF THE EIGHTH-ORDER BANDPASS FILTER (120Hz) IN CONFIGURATION (a) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	v_{n-1}^j	v_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 0 1 0	1 0 0 1 1 1 0 1
0	0	1	1	0	16	0 0 0 1 0 0 1 1	0 1 1 1 0 0 0 1
0	1	0	0	0	L	1 1 1 0 0 1 1 0	1 0 1 1 0 0 1 0
0	1	0	1	0	O	1 1 0 0 0 1 1 1	1 0 0 0 0 1 1 0
0	1	1	0	0	C	0 0 0 1 1 0 0 1	0 1 0 0 1 1 1 1
0	1	1	1	0	A	1 1 1 1 1 0 1 0	0 0 1 0 0 0 1 1
1	0	0	0	0	T	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0
1	0	0	1	0	I	0 0 0 0 0 0 0 0	1 1 0 1 0 1 0 0
1	0	1	0	0	O	0 1 0 1 0 0 1 0	1 0 0 1 1 1 0 1
1	0	1	1	0	N	0 0 1 1 0 0 1 1	0 1 1 1 0 0 0 1
1	1	0	0	0	S	0 0 0 0 0 1 1 0	1 0 1 1 0 0 1 0
1	1	0	1	0		1 1 1 0 0 1 1 1	1 0 0 0 0 1 1 0
1	1	1	0	0		0 0 1 1 1 0 0 1	0 1 0 0 1 1 1 1
1	1	1	1	0	↓	0 0 0 1 1 0 1 0	0 0 1 0 0 0 1 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1		0 0 1 1 0 0 1 1	0 0 0 1 1 1 0 1
0	0	1	1	1	x_{max}	0 0 0 1 0 0 1 1	1 1 1 1 0 0 0 1
0	1	0	0	1	L	1 1 1 0 0 1 1 1	0 0 1 1 0 0 1 0
0	1	0	1	1	O	1 1 0 0 1 0 0 0	0 0 0 0 0 1 1 0
0	1	1	0	1	C	0 0 0 1 1 0 0 1	1 1 0 0 1 1 1 1
0	1	1	1	1	A	1 1 1 1 1 0 1 0	1 0 1 0 0 0 1 1
1	0	0	0	1	T	0 0 1 0 0 0 0 0	1 0 0 0 0 0 0 0
1	0	0	1	1	I	0 0 0 0 0 0 0 1	0 1 0 1 0 1 0 0
1	0	1	0	1	O	0 1 0 1 0 0 1 1	0 0 0 1 1 1 0 1
1	0	1	1	1	N	0 0 1 1 0 0 1 1	1 1 1 1 0 0 0 1
1	1	0	0	1	S	0 0 0 0 0 1 1 1	0 0 1 1 0 0 1 0
1	1	0	1	1		1 1 1 0 1 0 0 0	0 0 0 0 0 1 1 0
1	1	1	0	1		0 0 1 1 1 0 0 1	1 1 0 0 1 1 1 1
1	1	1	1	1	↓	0 0 0 1 1 0 1 0	1 0 1 0 0 0 1 1

TABLE C7: CBFCS MEMORY FOR THE REALIZATION OF THE SECOND PASSBAND ELEMENTAL FILTER $H_2^P(z)$ OF THE EIGHTH-ORDER BANDPASS FILTER (120Hz) IN CONFIGURATION (a) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	u_{n-1}^j	u_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 0 1 0	0 0 0 1 1 1 0 1
0	0	1	1	0	16	0 0 0 1 0 0 1 0	1 1 1 1 0 0 0 1
0	1	0	0	0	L	0 0 0 0 1 1 0 0	1 0 0 0 1 0 0 0
0	1	0	1	0	0	1 1 1 0 1 1 0 1	0 1 0 1 1 1 0 0
0	1	1	0	0	C	0 0 1 1 1 1 1 0	1 0 1 0 0 1 0 1
0	1	1	1	0	A	0 0 0 1 1 1 1 1	0 1 1 1 1 0 0 1
1	0	0	0	0	T	1 1 1 1 0 0 0 0	0 0 0 0 0 0 0 0
1	0	0	1	0	I	1 1 0 1 0 0 0 0	1 1 0 1 0 1 0 0
1	0	1	0	0	0	0 0 1 0 0 0 1 0	0 0 0 1 1 1 0 1
1	0	1	1	0	N	0 0 0 0 0 0 1 0	1 1 1 1 0 0 0 1
1	1	0	0	0	S	1 1 1 1 1 1 0 0	1 0 0 0 1 0 0 0
1	1	0	1	0		1 1 0 1 1 1 0 1	0 1 0 1 1 1 0 0
1	1	1	0	0		0 0 1 0 1 1 1 0	1 0 1 0 0 1 0 1
1	1	1	1	0	↓	0 0 0 0 1 1 1 1	0 1 1 1 1 0 0 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1		0 0 1 1 0 0 1 0	1 0 0 1 1 1 0 1
0	0	1	1	1	x_{max}	0 0 0 1 0 0 1 1	0 1 1 1 0 0 0 1
0	1	0	0	1	L	0 0 0 0 1 1 0 1	0 0 0 0 1 0 0 0
0	1	0	1	1	O	1 1 1 0 1 1 0 1	1 1 0 1 1 1 0 0
0	1	1	0	1	C	0 0 1 1 1 1 1 1	0 0 1 0 0 1 0 1
0	1	1	1	1	A	0 0 0 1 1 1 1 1	1 1 1 1 1 0 0 1
1	0	0	0	1	T	1 1 1 1 0 0 0 0	1 0 0 0 0 0 0 0
1	0	0	1	1	I	1 1 0 1 0 0 0 1	0 1 0 1 0 1 0 0
1	0	1	0	1	O	0 0 1 0 0 0 1 0	1 0 0 1 1 1 0 1
1	0	1	1	1	N	0 0 0 0 0 0 1 1	0 1 1 1 0 0 0 1
1	1	0	0	1	S	1 1 1 1 1 1 0 1	0 0 0 0 1 0 0 0
1	1	0	1	1		1 1 0 1 1 1 0 1	1 1 0 1 1 1 0 0
1	1	1	0	1		0 0 1 0 1 1 1 1	0 0 1 0 0 1 0 1
1	1	1	1	1	↓	0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 1

TABLE C8: CBFCS MEMORY FOR THE REALIZATION OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER $-H_u^T(z)$ WITH TRANSITION SAMPLE VALUE $W_n = 0.5$ IN CONFIGURATION (a) FOR THE EIGHTH-ORDER BANDPASS FILTER (120Hz) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	y_{n-1}^j	y_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 1 0 0	0 0 0 0 1 1 1 1
0	0	1	1	0	16	0 0 0 1 0 1 0 0	1 1 1 0 0 0 1 1
0	1	0	0	0	L	1 1 1 1 0 1 0 1	0 0 1 1 0 0 1 1
0	1	0	1	0	O	1 1 0 1 0 1 1 0	0 0 0 0 0 1 1 1
0	1	1	0	0	C	0 0 1 0 1 0 0 1	0 1 0 0 0 0 1 0
0	1	1	1	0	A	0 0 0 0 1 0 1 0	0 0 0 1 0 1 1 0
1	0	0	0	0	T	0 0 0 0 1 1 0 1	0 1 0 0 0 1 1 1
1	0	0	1	0	I	1 1 1 0 1 1 1 0	0 0 0 1 1 0 1 1
1	0	1	0	0	O	0 1 0 0 0 0 0 1	0 1 0 1 0 1 1 0
1	0	1	1	0	N	0 0 1 0 0 0 1 0	0 0 1 0 1 0 1 0
1	1	0	0	0	S	0 0 0 0 0 0 1 0	0 1 1 1 1 0 1 0
1	1	0	1	0		1 1 1 0 0 0 1 1	0 1 0 0 1 1 1 0
1	1	1	0	0		0 0 1 1 0 1 1 0	1 0 0 0 1 0 0 1
1	1	1	1	0	↓	0 0 0 1 0 1 1 1	0 1 0 1 1 1 0 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1		0 0 1 1 0 1 0 0	1 0 0 0 1 1 1 1
0	0	1	1	1	x_{max}	0 0 0 1 0 1 0 1	0 1 1 0 0 0 1 1
0	1	0	0	1	L	1 1 1 1 0 1 0 1	1 0 1 1 0 0 1 1
0	1	0	1	1	O	1 1 0 1 0 1 1 0	1 0 0 0 0 1 1 1
0	1	1	0	1	C	0 0 1 0 1 0 0 1	1 1 0 0 0 0 1 0
0	1	1	1	1	A	0 0 0 0 1 0 1 0	1 0 0 1 0 1 1 0
1	0	0	0	1	T	0 0 0 0 1 1 0 1	1 1 0 0 0 1 1 1
1	0	0	1	1	I	1 1 1 0 1 1 1 0	1 0 0 1 1 0 1 1
1	0	1	0	1	O	0 1 0 0 0 0 0 1	1 1 0 1 0 1 1 0
1	0	1	1	1	N	0 0 1 0 0 0 1 0	1 0 1 0 1 0 1 0
1	1	0	0	1	S	0 0 0 0 0 0 1 0	1 1 1 1 1 0 1 0
1	1	0	1	1		1 1 1 0 0 0 1 1	1 1 0 0 1 1 1 0
1	1	1	0	1		0 0 1 1 0 1 1 1	0 0 0 0 1 0 0 1
1	1	1	1	1	↓	0 0 0 1 0 1 1 1	1 1 0 1 1 1 0 1

TABLE C9: CBFCS MEMORY FOR THE REALIZATION OF THE LOWER TRANSITION SECOND-ORDER ELEMINTAL FILTER $H_1^T(z)$ WITH TRANSITION SAMPLE VALUE W_n=0.4149780277 IN CONFIGURATION (a) FOR THE EIGHTH-ORDER BANDPASS FILTER (120Hz) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	w_{n-1}^j	w_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0				0 0 0 0 0 0 0 0					
0	0	0	1	0		1 1 1 0 0 0 0 0				1 1 0 1 0 1 0 0					
0	0	1	0	0		0 0 1 1 0 0 1 1				1 0 0 1 0 1 1 0					
0	0	1	1	0	16	0 0 0 1 0 1 0 0				0 1 1 1 1 0 1 0					
0	1	0	0	0	L	0 0 0 1 1 0 0 1				1 1 0 0 1 0 1 1					
0	1	0	1	0	0	1 1 1 1 1 0 1 0				1 0 0 1 1 1 1 1					
0	1	1	0	0	C	0 1 0 0 1 1 0 1				0 1 1 0 0 0 0 1					
0	1	1	1	0	A	0 0 1 0 1 1 1 0				0 0 1 1 0 1 0 1					
1	0	0	0	0	T	1 1 1 0 0 0 0 0				0 0 0 0 0 0 0 0					
1	0	0	1	0	I	1 1 0 0 0 0 0 0				1 1 0 1 0 1 0 0					
1	0	1	0	0	O	0 0 0 1 0 0 1 1				1 0 0 1 0 1 1 0					
1	0	1	1	0	N	1 1 1 1 0 1 0 0				0 1 1 1 1 0 1 0					
1	1	0	0	0	S	1 1 1 1 1 0 0 1				1 1 0 0 1 0 1 1					
1	1	0	1	0	↓	1 1 0 1 1 0 1 0				1 0 0 1 1 1 1 1					
1	1	1	0	0		0 0 1 0 1 1 0 1				0 1 1 0 0 0 0 1					
1	1	1	1	0		0 0 0 0 1 1 1 0				0 0 1 1 0 1 0 1					
0	0	0	0	1	↑	0 0 0 0 0 0 0 0				1 0 0 0 0 0 0 0					
0	0	0	1	1		1 1 1 0 0 0 0 1				0 1 0 1 0 1 0 0					
0	0	1	0	1		0 0 1 1 0 1 0 0				0 0 0 1 0 1 1 0					
0	0	1	1	1	χ_{max}	0 0 0 1 0 1 0 0				1 1 1 1 1 0 1 0					
0	1	0	0	1	L	0 0 0 1 1 0 1 0				0 1 0 0 1 0 1 1					
0	1	0	1	1	O	1 1 1 1 1 0 1 1				0 0 0 1 1 1 1 1					
0	1	1	0	1	C	0 1 0 0 1 1 0 1				1 1 1 0 0 0 0 1					
0	1	1	1	1	A	0 0 1 0 1 1 1 0				1 0 1 1 0 1 0 1					
1	0	0	0	1	T	1 1 1 0 0 0 0 0				1 0 0 0 0 0 0 0					
1	0	0	1	1	I	1 1 0 0 0 0 0 1				0 1 0 1 0 1 0 0					
1	0	1	0	1	O	0 0 0 1 0 1 0 0				0 0 0 1 0 1 1 0					
1	0	1	1	1	N	1 1 1 1 0 1 0 0				1 1 1 1 1 0 1 0					
1	1	0	0	1	S	1 1 1 1 1 0 1 0				0 1 0 0 1 0 1 1					
1	1	0	1	1	↓	1 1 0 1 1 0 1 1				0 0 0 1 1 1 1 1					
1	1	1	0	1		0 0 1 0 1 1 0 1				1 1 1 0 0 0 0 1					
1	1	1	1	1		0 0 0 0 1 1 1 0				1 0 1 1 0 1 0 1					

TABLE C10: CBFCS MEMORY FOR THE REALIZATION OF THE FIRST PASSBAND ELEMENTAL FILTER $-H_1^P(z)$ OF THE EIGHTH-ORDER BANDPASS FILTER (120Hz) IN CONFIGURATION (a) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	v_{n-1}^j	v_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 0 1 0	1 0 0 1 1 1 0 1
0	0	1	1	0	16	0 0 0 1 0 0 1 1	0 1 1 1 0 0 0 1
0	1	0	0	0	L	1 1 1 0 0 1 1 0	1 0 1 1 0 0 1 0
0	1	0	1	0	O	1 1 0 0 0 1 1 1	1 0 0 0 0 1 1 0
0	1	1	0	0	C	0 0 0 1 1 0 0 1	0 1 0 0 1 1 1 1
0	1	1	1	0	A	1 1 1 1 1 0 1 0	0 0 1 0 0 0 1 1
1	0	0	0	0	T	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0
1	0	0	1	0	I	0 0 0 0 0 0 0 0	1 1 0 1 0 1 0 0
1	0	1	0	0	O	0 1 0 1 0 0 1 0	1 0 0 1 1 1 0 1
1	0	1	1	0	N	0 0 1 1 0 0 1 1	0 1 1 1 0 0 0 1
1	1	0	0	0	S	0 0 0 0 0 1 1 0	1 0 1 1 0 0 1 0
1	1	0	1	0		1 1 1 0 0 1 1 1	1 0 0 0 0 1 1 0
1	1	1	0	0		0 0 1 1 1 0 0 1	0 1 0 0 1 1 1 1
1	1	1	1	0	↓	0 0 0 1 1 0 1 0	0 0 1 0 0 0 1 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1		0 0 1 1 0 0 1 1	0 0 0 1 1 1 0 1
0	0	1	1	1	x_{max}	0 0 0 1 0 0 1 1	1 1 1 1 0 0 0 1
0	1	0	0	1	L	1 1 1 0 0 1 1 1	0 0 1 1 0 0 1 0
0	1	0	1	1	O	1 1 0 0 1 0 0 0	0 0 0 0 0 1 1 0
0	1	1	0	1	C	0 0 0 1 1 0 0 1	1 1 0 0 1 1 1 1
0	1	1	1	1	A	1 1 1 1 1 0 1 0	1 0 1 0 0 0 1 1
1	0	0	0	1	T	0 0 1 0 0 0 0 0	1 0 0 0 0 0 0 0
1	0	0	1	1	I	0 0 0 0 0 0 0 1	0 1 0 1 0 1 0 0
1	0	1	0	1	O	0 1 0 1 0 0 1 1	0 0 0 1 1 1 0 1
1	0	1	1	1	N	0 0 1 1 0 0 1 1	1 1 1 1 0 0 0 1
1	1	0	0	1	S	0 0 0 0 0 1 1 1	0 0 1 1 0 0 1 0
1	1	0	1	1		1 1 1 0 1 0 0 0	0 0 0 0 0 1 1 0
1	1	1	0	1		0 0 1 1 1 0 0 1	1 1 0 0 1 1 1 1
1	1	1	1	1	↓	0 0 0 1 1 0 1 0	1 0 1 0 0 0 1 1

TABLE C11: CBFCS MEMORY FOR THE REALIZATION OF THE SECOND PASSBAND ELEMENTAL FILTER $H_2^P(z)$ OF THE EIGHTH-ORDER BANDPASS FILTER (120Hz) IN CONFIGURATION (a) SHOWN IN FIGURE 8.1.2

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	u_{n-1}^j	u_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 1 0 0 0 0 0	1 1 0 1 0 1 0 0
0	0	1	0	0		0 0 1 1 0 1 0 0	0 0 0 0 1 1 1 1
0	0	1	1	0	16	0 0 0 1 0 1 0 0	1 1 1 0 0 0 1 1
0	1	0	0	0	L	0 0 0 0 1 0 1 0	0 1 1 0 0 1 1 0
0	1	0	1	0	0	1 1 1 0 1 0 1 1	0 0 1 1 1 0 1 0
0	1	1	0	0	C	0 0 1 1 1 1 1 0	0 1 1 1 0 1 0 1
0	1	1	1	0	A	0 0 0 1 1 1 1 1	0 1 0 0 1 0 0 1
1	0	0	0	0	T	1 1 1 1 0 0 1 0	1 0 1 1 1 0 0 1
1	0	0	1	0	I	1 1 0 1 0 0 1 1	1 0 0 0 1 1 0 1
1	0	1	0	0	0	0 0 1 0 0 1 1 0	1 1 0 0 1 0 0 0
1	0	1	1	0	N	0 0 0 0 0 1 1 1	1 0 0 1 1 1 0 0
1	1	0	0	0	S	1 1 1 1 1 1 0 1	0 0 0 1 1 1 1 1
1	1	0	1	0	|	1 1 0 1 1 1 0 1	1 1 1 1 0 0 1 1
1	1	1	0	0	|	0 0 1 1 0 0 0 1	0 0 1 0 1 1 1 0
1	1	1	1	0	↓	0 0 0 1 0 0 1 0	0 0 0 0 0 0 1 0
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1	|	1 1 1 0 0 0 0 1	0 1 0 1 0 1 0 0
0	0	1	0	1	|	0 0 1 1 0 1 0 0	1 0 0 0 1 1 1 1
0	0	1	1	1	x_{max}	0 0 0 1 0 1 0 1	0 1 1 0 0 0 1 1
0	1	0	0	1	L	0 0 0 0 1 0 1 0	1 1 1 0 0 1 1 0
0	1	0	1	1	0	1 1 1 0 1 0 1 1	1 0 1 1 1 0 1 0
0	1	1	0	1	C	0 0 1 1 1 1 1 0	1 1 1 1 0 1 0 1
0	1	1	1	1	A	0 0 0 1 1 1 1 1	1 1 0 0 1 0 0 1
1	0	0	0	1	T	1 1 1 1 0 0 1 1	0 0 1 1 1 0 0 1
1	0	0	1	1	I	1 1 0 1 0 1 0 0	0 0 0 0 1 1 0 1
1	0	1	0	1	O	0 0 1 0 0 1 1 1	0 1 0 0 1 0 0 0
1	0	1	1	1	N	0 0 0 0 1 0 0 0	0 0 0 1 1 1 0 0
1	1	0	0	1	S	1 1 1 1 1 1 0 1	1 0 0 1 1 1 1 1
1	1	0	1	1	|	1 1 0 1 1 1 1 0	0 1 1 1 0 0 1 1
1	1	1	0	1	|	0 0 1 1 0 0 0 1	1 0 1 0 1 1 1 0
1	1	1	1	1	↓	0 0 0 1 0 0 1 0	1 0 0 0 0 0 1 0

TABLE C12: CBFCS MEMORY FOR THE REALIZATION OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER-$H_u^T(z)$ WITH TRANSITION SAMPLE VALUE W_n = 0.4149780277 IN CONFIGURATION (a) FOR THE EIGHTH-ORDER BANDPASS FILTER (120Hz) SHOWN IN FIGURE 8.1.2

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	y_{n-1}^j	y_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0		1	1	0	0	0	0	0	0	0	1
0	0	1	0	0		1	1	0	0	0	0	0	0	1	0
0	0	1	1	0	16	1	0	0	0	0	0	0	0	1	0
0	1	0	0	0	L	0	0	0	1	1	1	1	1	1	1
0	1	0	1	0	0	1	1	0	1	1	1	1	1	1	1
0	1	1	0	0	C	1	1	1	0	0	0	0	0	0	0
0	1	1	1	0	A	1	0	1	0	0	0	0	0	0	1
1	0	0	0	0	T	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	I	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	O	0	0	0	0	0	0	0	0	1	0
1	0	1	1	0	N	1	1	0	0	0	0	0	0	1	0
1	1	0	0	0	S	0	1	0	1	1	1	1	1	1	0
1	1	0	1	0	|	0	0	0	1	1	1	1	1	1	1
1	1	1	0	0	|	0	0	1	0	0	0	0	0	1	0
1	1	1	1	0	↓	1	1	1	0	0	0	0	0	1	1
0	0	0	0	1	↑	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1		1	1	0	0	0	0	0	0	0	1
0	0	1	0	1		1	1	0	0	0	0	0	0	1	0
0	0	1	1	1	χ_{max}	1	0	0	0	0	0	0	0	1	0
0	1	0	0	1	L	0	0	1	0	0	0	0	0	0	1
0	1	0	1	1	O	1	1	1	0	0	0	0	0	0	1
0	1	1	0	1	C	1	1	1	0	0	0	0	0	1	0
0	1	1	1	1	A	1	0	1	0	0	0	0	0	1	1
1	0	0	0	1	T	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	I	0	0	0	0	0	0	0	0	0	1
1	0	1	0	1	O	0	0	0	0	0	0	0	0	1	0
1	0	1	1	1	N	1	1	0	0	0	0	0	0	1	0
1	1	0	0	1	S	0	1	1	0	0	0	0	0	1	0
1	1	0	1	1	|	0	0	1	0	0	0	0	0	1	1
1	1	1	0	1	|	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	↓	1	1	1	0	0	0	0	0	1	1

TABLE C13: CBFCS MEMORY FOR THE REALIZATION OF THE LOWER TRANSITION SECOND-ORDER ELEMENTAL FILTER $H_l^T(z)$ WITH TRANSITION SAMPLE VALUE $W_n = 0.453125$ IN CONFIGURATION (b) FOR THE EIGHTH-ORDER BANDPASS FILTER (1MHz) SHOWN IN FIGURE 8.1.2

MEMORY ADDRESS						MEMORY CONTENT											
x_n^j	x_{n-1}^j	w_{n-1}^j	w_{n-2}^j	$C_{8,n}^{j+1}$		MSB											LSB
0	0	0	0	0	↑	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0		1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0		1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	1	1	0	16	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	L	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	0	O	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	C	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	1	1	0	A	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	T	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	1	1	0	N	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	S	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	0		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0		0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	↓	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	↑	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1		1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	1	0	1		1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	1	1	1	x_{max}	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
0	1	0	0	1	L	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1
0	1	0	1	1	O	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	1	0	1	C	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0
0	1	1	1	1	A	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1
1	0	0	0	1	T	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	I	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
1	0	1	0	1	O	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
1	0	1	1	1	N	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0
1	1	0	0	1	S	0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	1	0	1	1		0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	1	0	1		0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	↓	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1

TABLE C14: CBFCS MEMORY FOR THE REALIZATION OF THE FIRST PASSBAND ELEMENTAL FILTER $-H_1^P(z)$ OF THE EIGHTH-ORDER BANDPASS FILTER (1MHz) IN CONFIGURATION (b) SHOWN IN FIGURE 8.1.2.

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	v_{n-1}^j	v_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0								0 0 0 0 0 0 0 0	
0	0	0	1	0		1 1 0 0 0 0 0 0								0 0 0 0 0 0 0 1	
0	0	1	0	0		1 1 0 0 0 0 0 0								0 0 0 0 0 0 0 0	
0	0	1	1	0	16	1 0 0 0 0 0 0 0								0 0 0 0 0 0 0 1	
0	1	0	0	0	L	0 0 1 0 0 0 0 0								0 0 0 0 0 0 0 0	
0	1	0	1	0	O	1 1 1 0 0 0 0 0								0 0 0 0 0 0 0 1	
0	1	1	0	0	C	1 1 1 0 0 0 0 0								0 0 0 0 0 0 0 0	
0	1	1	1	0	A	1 0 1 0 0 0 0 0								0 0 0 0 0 0 0 1	
1	0	0	0	0	T	0 1 0 0 0 0 0 0								0 0 0 0 0 0 0 0	
1	0	0	1	0	I	0 0 0 0 0 0 0 0								0 0 0 0 0 0 0 1	
1	0	1	0	0	O	0 0 0 0 0 0 0 0								0 0 0 0 0 0 0 0	
1	0	1	1	0	N	1 1 0 0 0 0 0 0								0 0 0 0 0 0 0 1	
1	1	0	0	0	S	0 1 1 0 0 0 0 0								0 0 0 0 0 0 0 0	
1	1	0	1	0		0 0 1 0 0 0 0 0								0 0 0 0 0 0 0 1	
1	1	1	0	0		0 0 1 0 0 0 0 0								0 0 0 0 0 0 0 0	
1	1	1	1	0	↓	1 1 1 0 0 0 0 0								0 0 0 0 0 0 0 1	
0	0	0	0	1	↑	0 0 0 0 0 0 0 0								1 0 0 0 0 0 0 0	
0	0	0	1	1		1 1 0 0 0 0 0 0								1 0 0 0 0 0 0 1	
0	0	1	0	1		1 1 0 0 0 0 0 0								1 0 0 0 0 0 0 0	
0	0	1	1	1	x_{max}	1 0 0 0 0 0 0 0								1 0 0 0 0 0 0 1	
0	1	0	0	1	L	0 0 1 0 0 0 0 0								1 0 0 0 0 0 0 0	
0	1	0	1	1	O	1 1 1 0 0 0 0 0								1 0 0 0 0 0 0 1	
0	1	1	0	1	C	1 1 1 0 0 0 0 0								1 0 0 0 0 0 0 0	
0	1	1	1	1	A	1 0 1 0 0 0 0 0								1 0 0 0 0 0 0 1	
1	0	0	0	1	T	0 1 0 0 0 0 0 0								1 0 0 0 0 0 0 0	
1	0	0	1	1	I	0 0 0 0 0 0 0 0								1 0 0 0 0 0 0 1	
1	0	1	0	1	O	0 0 0 0 0 0 0 0								1 0 0 0 0 0 0 0	
1	0	1	1	1	N	1 1 0 0 0 0 0 0								1 0 0 0 0 0 0 1	
1	1	0	0	1	S	0 1 1 0 0 0 0 0								1 0 0 0 0 0 0 0	
1	1	0	1	1		0 0 1 0 0 0 0 0								1 0 0 0 0 0 0 1	
1	1	1	0	1		0 0 1 0 0 0 0 0								1 0 0 0 0 0 0 0	
1	1	1	1	1	↓	1 1 1 0 0 0 0 0								1 0 0 0 0 0 0 1	

TABLE C15: CBFCS MEMORY FOR THE REALIZATION OF THE SECOND PASSBAND ELEMENTAL FILTER $H_2^P(z)$ OF THE EIGHTH-ORDER BANDPASS FILTER (1MHz) IN CONFIGURATION (b) SHOWN IN FIGURE 8.1.2

MEMORY ADDRESS						MEMORY CONTENT									
x_n^j	x_{n-1}^j	u_{n-1}^j	u_{n-2}^j	$C_{8,n}^{j+1}$		MSB									LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0				0 0 0 0 0 0 0 0					
0	0	0	1	0		1 1 0 0 0 0 0 0				0 0 0 0 0 0 0 1					
0	0	1	0	0		1 0 1 1 1 1 1 1				1 1 1 1 1 1 1 0					
0	0	1	1	0	16	1 0 0 0 0 0 0 0				0 0 0 0 0 0 0 0					
0	1	0	0	0	L	0 0 1 0 0 0 0 0				0 0 0 0 0 0 0 1					
0	1	0	1	0	0	1 1 1 0 0 0 0 0				0 0 0 0 0 0 1 0					
0	1	1	0	0	C	1 1 0 1 1 1 1 1				1 1 1 1 1 1 1 1					
0	1	1	1	0	A	1 0 1 0 0 0 0 0				0 0 0 0 0 0 0 0					
1	0	0	0	0	T	0 1 0 0 0 0 0 0				0 0 0 0 0 0 0 0					
1	0	0	1	0	I	0 0 0 0 0 0 0 0				0 0 0 0 0 0 0 1					
1	0	1	0	0	O	1 1 1 1 1 1 1 1				1 1 1 1 1 1 1 0					
1	0	1	1	0	N	1 0 1 1 1 1 1 1				1 1 1 1 1 1 1 1					
1	1	0	0	0	S	0 1 1 0 0 0 0 0				0 0 0 0 0 0 0 1					
1	1	0	1	0	↓	0 0 1 0 0 0 0 0				0 0 0 0 0 0 1 0					
1	1	1	0	0		0 0 0 1 1 1 1 1				1 1 1 1 1 1 1 1					
1	1	1	1	0		1 1 1 0 0 0 0 0				0 0 0 0 0 0 0 0					
0	0	0	0	1	↑	0 0 0 0 0 0 0 0				1 0 0 0 0 0 0 0					
0	0	0	1	1		1 1 0 0 0 0 0 0				1 0 0 0 0 0 0 1					
0	0	1	0	1		1 1 0 0 0 0 0 0				0 1 1 1 1 1 1 0					
0	0	1	1	1	x_{max}	1 0 0 0 0 0 0 0				0 1 1 1 1 1 1 1					
0	1	0	0	1	L	0 0 1 0 0 0 0 0				1 0 0 0 0 0 0 1					
0	1	0	1	1	0	1 1 1 0 0 0 0 0				1 0 0 0 0 0 1 0					
0	1	1	0	1	C	1 1 1 0 0 0 0 0				0 1 1 1 1 1 1 1					
0	1	1	1	1	A	1 0 1 0 0 0 0 0				1 0 0 0 0 0 0 0					
1	0	0	0	1	T	0 1 0 0 0 0 0 0				1 0 0 0 0 0 0 0					
1	0	0	1	1	I	0 0 0 0 0 0 0 0				1 0 0 0 0 0 0 1					
1	0	1	0	1	O	0 0 0 0 0 0 0 0				0 1 1 1 1 1 1 0					
1	0	1	1	1	N	1 1 0 0 0 0 0 0				0 1 1 1 1 1 1 1					
1	1	0	0	1	S	0 1 1 0 0 0 0 0				1 0 0 0 0 0 0 1					
1	1	0	1	1	↓	0 0 1 0 0 0 0 0				1 0 0 0 0 0 1 0					
1	1	1	0	1		0 0 1 0 0 0 0 0				0 1 1 1 1 1 1 1					
1	1	1	1	1		1 1 1 0 0 0 0 0				1 0 0 0 0 0 0 0					

TABLE C16: CBFCS MEMORY FOR THE REALIZATION OF THE UPPER TRANSITION SECOND-ORDER ELEMENTAL FILTER $-H_u^T(z)$ WITH TRANSITION SAMPLE VALUE $W_n = 0.453125$ IN CONFIGURATION (b) FOR THE EIGHTH-ORDER BANDPASS FILTER (1MHz) SHOWN IN FIGURE 8.1.2

MEMORY ADDRESS						MEMORY CONTENT	
x_n^j	x_{n-1}^j	y_{n-1}^j	y_{n-2}^j	$C_{8,n}^{j+1}$		MSB	LSB
0	0	0	0	0	↑	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0	0	0	1	0		1 1 0 0 0 1 1 0	0 1 0 0 0 0 1 0
0	0	1	0	0		1 1 0 0 0 0 1 1	0 0 1 0 1 1 1 1
0	0	1	1	0	16	1 0 0 0 1 0 0 1	0 1 1 1 0 0 0 1
0	1	0	0	0	L	0 0 0 1 1 1 1 0	0 1 1 0 1 0 0 0
0	1	0	1	0	O	1 1 1 0 0 1 0 0	1 0 1 0 1 0 1 0
0	1	1	0	0	C	1 1 1 0 0 0 0 1	1 0 0 1 0 1 1 1
0	1	1	1	0	A	1 0 1 0 0 1 1 1	1 1 0 1 1 0 0 1
1	0	0	0	0	T	0 0 1 1 1 0 0 1	1 0 0 1 1 0 1 0
1	0	0	1	0	I	1 1 1 1 1 1 1 1	1 1 0 1 1 1 0 0
1	0	1	0	0	O	1 1 1 1 1 1 0 0	1 1 0 0 1 0 0 1
1	0	1	1	0	N	1 1 0 0 0 0 1 1	0 0 0 0 1 0 1 1
1	1	0	0	0	S	0 1 0 1 1 0 0 0	0 0 0 0 0 0 1 0
1	1	0	1	0	↓	0 0 0 1 1 1 1 0	0 1 0 0 0 1 0 0
1	1	1	0	0		0 0 0 1 1 0 1 1	0 0 1 1 0 0 0 1
1	1	1	1	0		1 1 1 0 0 0 0 1	0 1 1 1 0 0 1 1
0	0	0	0	1	↑	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
0	0	0	1	1		1 1 0 0 0 1 1 0	1 1 0 0 0 0 1 0
0	0	1	0	1		1 1 0 0 0 0 1 1	1 0 1 0 1 1 1 1
0	0	1	1	1	x_{max}	1 0 0 0 1 0 0 1	1 1 1 1 0 0 0 1
0	1	0	0	1	L	0 0 0 1 1 1 1 0	1 1 1 0 1 0 0 0
0	1	0	1	1	O	1 1 1 0 0 1 0 1	0 0 1 0 1 0 1 0
0	1	1	0	1	C	1 1 1 0 0 0 1 0	0 0 0 1 0 1 1 1
0	1	1	1	1	A	1 0 1 0 1 0 0 0	0 1 0 1 1 0 0 1
1	0	0	0	1	T	0 0 1 1 1 0 1 0	0 0 0 1 1 0 1 0
1	0	0	1	1	I	0 0 0 0 0 0 0 0	0 1 0 1 1 1 0 0
1	0	1	0	1	O	1 1 1 1 1 1 0 1	0 1 0 0 1 0 0 1
1	0	1	1	1	N	1 1 0 0 0 0 1 1	1 0 0 0 1 0 1 1
1	1	0	0	1	S	0 1 0 1 1 0 0 0	1 0 0 0 0 0 1 0
1	1	0	1	1	↓	0 0 0 1 1 1 1 0	1 1 0 0 0 1 0 0
1	1	1	0	1		0 0 0 1 1 0 1 1	1 0 1 1 0 0 0 1
1	1	1	1	1		1 1 1 0 0 0 0 1	1 1 1 1 0 0 1 1

TABLE C17: CBFCS MEMORY FOR A SECOND-ORDER ELEMENTAL FILTER CENTRED AT 1000100Hz.

REFERENCES

(1) Stuart Lawson.
"Theory and design of digital filters".
Electronic Engineering, May 1977, p43-46.

(2) Martin Terrell.
"The emerging world of digital filters".
Electronic Engineering, Oct. 1977, p26-31.

(3) Gwyn Edwards.
"Digital filters can simplify signal processing".
Electronic Engineering, June 1976, p53-55.

(4) C.R.Rabiner, L.B.Jackson, R.W.Schafer, C.H.Coker.
"A hardware realization of a digital formant speech synthesizer".
IEEE Trans., Vol: COM-19, No: 6, Dec. 1971, p1016-1020.

(5) ISSCC Wrap-up.
"Ultra-high-speed logic: more than just a promise in LSI".
Electronic Design, No: 7, March 1977, p34-35.

(6) Samuel Derman.
"Progress in gigabit logic reported for super-fast switching uses".
Electronic Design, No: 15, July 1976, p34-38.

(7) Fritz Meyer.
"BCL: a high-speed logic technology".
Electronic Engineering, Sept. 1977 p79-81.

(8) P.F.Adams.
"The use of digital filters in telecommunications".
Electronic Engineering, April 1977, p45-46.

(9) L.R.Rabiner et.al.
"Terminology in digital signal processing".
IEEE Trans., Vol: AU-20, No: 5, Dec. 1972, p322-337.

(10) L.R.Rabiner, B.Gold.
"Theory and application of digital signal processing".
Prentice-Hall, 1975.

(11) A.V.Oppenheim, R.W.Schafer.
"Digital signal Processing".
Prentice-Hall, 1975.

(12) D.K.Cheng.
"Analysis of linear system".
Addison-Wesley, 1960, p321-322.

(13) S.A.Tretter.
"Introduction to discrete-time signal processing".
John-Wiley & Son, 1976, chapters 1 and 2.

(14) J.F.Kaiser.
"Digital Filters".
Chapter 7 in "System analysis by digital computer" by F.F.Kuo and J.F.Kaiser. John-Wiley & Son, 1966.

(15) R.M.Golden, J.F.Kaiser.
"Design of wideband sampled-data filters".
B.S.T.J. Vol: 43, No: 2, Feb. 1967, p149-171.

(16) A.G.Constantinides.
"Frequency transformations for digital filters".
Electronic letters, No: 3, 1967, p487-489.

(17) A.G.Constantinides.
"Frequency transformations for digital filters."
Electronic letters, Vol: 4, No: 7, April 1968, p115-116.

(18) A.G.Constantinides.
"Spectral transformations for digital filters".
IEE Proc., Vol:117, No: 8, Aug. 1970, p1585-1590.

(19) A.M.Bush, D.C.Fielder.
"Simplified algebra for the Bilinear and related transformations".
IEEE Trans., Vol: AU-21, April 1973, p127-128.

(20) C.S.Barrus, T.W.Parks.
"Time domain design of recursive digital filters".
IEEE Trans., Vol: AU-18, 1970, p137-141.

(21) F.Brophy, A.C.Salazar.
"Considerations of the Pade Approximant Technique in the synthesis of recursive digital filters".
IEEE Trans., Vol: AU-21, No: 6, 1973, p500-505.

(22) A.G.Evans, R.Fischl.
"Optimal least squares time-domain synthesis of recursive digital filters".
IEEE Trans., Vol: AU-21, No: 1, Feb. 1973, p61-65.

(23) F.Brophy, A.C.Salazar.
"Recursive digital filter synthesis in the time-domain".
IEEE Trans., Vol: ASSP-22, No: 1, Feb. 1974, p45-55.

(24) P.Thajchayapong, P.J.W.Rayner.
"Recursive digital filter design by linear programming".
IEEE Trans., Vol: AU-21, 1973, p107-112.

(25) L.R.Rabiner, N.Y.Graham, H.D.Helms.
"Linear programming design of IIR digital filters with arbitrary magnitude function".
IEEE Trans., Vol: ASSP-22, No: 2, April 1974, p117-123.

(26) R.Fletcher, M.J.D.Powell.
"A rapidly convergent descent method for minimization".
Computer J., 6, No: 2, 1963, p163-168.

(27) H.D.Helms.
"Non-recursive digital filters: design methods for achieving specifications on frequency response".
IEEE Trans., Vol: AU-16, No: 3, Sept. 1968, p336-342.

(28) B.Gold, K.Jordan.
"A note on digital filter synthesis".
IEEE Proc., Vol: 56, No: 10, Oct. 1968, p1717-1718.

(29) B.Gold, K.Jordan.
"A direct search procedure for designing finite duration impulse response filters".
IEEE Trans., Vol: AU-17, No: 1, March 1969, p33-36.

(30) L.R.Rabiner, B.Gold, C.A.McGonegal.
"An approach to the approximation problem for non-recursive digital filters".
IEEE Trans., Vol: AU-18, No: 2, June 1970, p83-106.

(31) L.R.Rabiner, K.Steiglitz.
"The design of wide-band recursive and non-recursive digital differentiators".
IEEE Trans., Vol: AU-18, No: 2, June 1970, p204-209.

(32) L.R.Rabiner, R.W.Schafer.
"Recursive and non-recursive realizations of digital filters designed by frequency sampling techniques".
IEEE Trans., Vol: AU-19, No: 3, Sept., 1971, p200-207.

(33) L.R.Rabiner, R.W.Schafer.
"Correction to the above reference (32)".
IEEE Trans., Vol: AU-20, No: 1, March 1972, p104-105.

(34) P.A.Lynn.
"Economic linear-phase recursive digital filters".
Electronic letters, Vol: 6, No: 5, 1970, p143-145.

(35) P.A.Lynn.
"Recursive digital filters with linear-phase characteristics".
Computer J., Vol: 15, No: 4, 1971, p337-342.

(36) J.G.Proakis.
"Adaptive digital filters for equalization of telephone channels".
IEEE Trans., Vol: AU-18, No: 2, June 1970, p195-199.

(37) P.Morse, H.Feshbach.
"Summation of series".
In "Methods of theoretical Physics", McGraw-Hill, Vol: 1, Sec., 4-5, p413-414, 474.

(38) A.Erdelyi. (Ed.)
"Higher Transcendental Functions".
McGraw-Hill, 1954, p15, i.b.i.d., p48, p35-37.

(39) E.C.Titchmarsh.
"The theory of functions".
Oxford University Press, Sec., 3-3, 1939, p114.

(40) G.Hadley.
"Linear programming".
Addison-Wesley, 1963.

(41) R.G.Gallager.
"Information theory and reliable communication".
Wiley, 1968.

(42) A.Papoulis.
"On the approximation problem in filter design".
IRE Conv. Rec., Pt: 2, 1957, p175-185.

(43) D.J.Wilde.
"Optimum seeking methods".
Prentice-Hall, 1964.

(44) C.M.Radar, B.Gold.
"Effects of parameter quantization on the poles of a digital filter".
IEEE Proc., May 1967, p688-689.

(45) Jackson, Kaiser, McDonald.
"An approach to the implementation of digital filters".
IEEE Trans., Vol: AU-16, No: 3, Sept. 1968, p413-421.

(46) R.A.Gabel.
"A parallel arithmetic hardware structure for recursive digital filtering".
IEEE Trans., Vol: ASSP-22, No: 4, 1974, p255-258.

(47) S.L.Freeny.
"Special purpose hardware for digital filtering".
IEEE Proc., Vol: 63, No: 4, April 1975, p633-648

(48) R.D.Mori.
"Cellular structures for implementing recursive and non-recursive digital filters".
The Radio & Electronic Engineer, Vol: 46, No: 4, April 1976, p173-181.

(49) C.R.Baugh, B.A.Wooley.
"A two's complement parallel array multiplication algorithm".
IEEE Trans., Vol: C-22, No: 12 Dec. 1973, p1045-1047.

(50) D.Kroft.
"Comments on 'A two's complement parallel array multiplication algorithm'".
IEEE Trans., Vol: C-23, Dec. 1974, p1327-1328.

(43) D.J.Wilde.
"Optimum seeking methods".
Prentice-Hall, 1964.

(44) C.M.Radar, B.Gold.
"Effects of parameter quantization on the poles of a digital filter".
IEEE Proc., May 1967, p688-689.

(45) Jackson, Kaiser, McDonald.
"An approach to the implementation of digital filters".
IEEE Trans., Vol: AU-16, No: 3, Sept. 1968, p413-421.

(46) R.A.Gabel.
"A parallel arithmetic hardware structure for recursive digital filtering".
IEEE Trans., Vol: ASSP-22, No: 4, 1974, p255-258.

(47) S.L.Freeny.
"Special purpose hardware for digital filtering".
IEEE Proc., Vol: 63, No: 4, April 1975, p633-648

(48) R.D.Mori.
"Cellular structures for implementing recursive and non-recursive digital filters".
The Radio & Electronic Engineer, Vol: 46, No: 4, April 1976, p173-181.

(49) C.R.Baugh, B.A.Wooley.
"A two's complement parallel array multiplication algorithm".
IEEE Trans., Vol: C-22, No: 12 Dec. 1973, p1045-1047.

(50) D.Kroft.
"Comments on 'A two's complement parallel array multiplication algorithm'".
IEEE Trans., Vol: C-23, Dec. 1974, p1327-1328.

(51) P.E.Blankenship.
"Comments on 'A two's complement parallel array multiplication algorithm'".
IEEE Trans., Vol: C-23, Dec. 1974, p1327.

(52) A.Habibi, P.A.Wintz.
"Fast multipliers"
IEEE Trans., Vol: C-19, Feb. 1970, p153-157.

(53) T.G.Hallin, M.J.Flynn.
"Pipelining of arithmetic functions".
IEEE Trans., Vol: C-21, Aug. 1972, p880-886.

(54) E.K.Cheng.
"A two's complemet pipeline multiplier".
IEEE International Conf. on ASSP, 1976, p647-650.

(55) R.F.Lyon.
"Two's complement pipeline multipliers".
IEEE Trans., Comm., April 1976, p418-425.

(56) M.A.Bin Nun, M.E.Woodward.
"A modular approach to the hardware implementation of digital filters".
The Radio & Electronic Engineer, Vol: 46, No: 8/9 Aug./Sept. 1976, p393-400.

(57) A.D.Booth.
"A signed binary multiplication technique".
Quart. J. Mech. Appl. Math., Vol: 4, Pt: 2, 1951.

(58) P.W.Baker.
"Parallel multiplicative algorithms for some elementary functions".
IEEE Trans., Vol: C-24, 1975 March, p322-325.

(59) L.P.Rubinfield.
"A proof of the modified Booth's algorithm for multiplication",
IEEE Trans., Vol: C-24, Oct. 1975, p1014-1015.

(60) O.L.MacSorley.
"High-speed arithmetic in binary computers".
IRE Proc., 1961, p67-91.

(61) P.M.Thompson, A.Belanger.
"Digital arithmetic units for a high data rate".
The Radio & electronic Engineer, Vol: 45, No: 3, March 1975,
p116-120.

(62) J.T.Quatse, R.A.Keir.
"A parallel accumulator for a general-purpose computer".
IEE Trans., Vol: 16, 1967, p165-171.

(63) C.W.Weller.
"A high-speed carry circuit for binary adders".
IEEE Trans., Vol: C-18, No: 8, Aug. 1969, p728-732.

(64) T.Lamdan, M.Ahnron.
"A circuit for high-speed carry propagation in LSI-FET
technology".
The Radio & Electronic Engineer, Vol: 46, No: 7, July 1976,
p337-341.

(65) C.S.Wallace.
"A suggestion for a fast multiplier".
IEEE Trans., Vol: EC-13, Feb. 1964, p14-17.

(66) C.I.Toma.
"Cellular logic array for high-speed signed binary number
multiplication".
IEEE Trans., Vol: C-24, Sept. 1975, p932-935.

(67) T.G.McDonald, R.K.Guha.
"The two's complement quasi-serial multipliers".
IEEE Trans., Vol: C-24, Dec. 1975, p1233-1235.

(68) Flores.
"The logic of computer arithmetic".
Prentice-Hall, 1963.

(69) B.E.Briley.
"Some new results on average worst case carry".
IEEE Trans., Vol: C-22, No: 5, May 1973, p459-463.

(70) A.D.Sypherd.
"Design of digital filters using ROM's".
Proc. National Electronic Conf., 1969, p691-693.

(71) A.Hemel.
"Making small ROM's do maths quickly, cheaply and easily".
Electronics International, May 1970, p104-111.

(72) H.Hellerman.
"Digital computer system principles".
McGraw-Hill, 1967.

(73) A.Peled, B.Liu, K.Steiglitz.
"A note on implementation of digital filters".
IEEE Trans., Vol: ASSP-23, Aug. 1975, p387-388.

(74) A.Peled, B.Liu.
"A new approach to the realization of non-recursive digital filters".
IEEE Trans., Vol: AU-21, No: 6, 1973, p477-484.

(75) A.Peled, B.Liu.
"Some new realizations of dedicated hardware digital signal processors".
IEEE Proc. EASCON, 1974, p464-468.

(76) A.Peled, B.Liu.
"A new hardware realization of digital filters".
IEEE Trans., Vol: ASSP-22, Dec. 1974, p456-462.

(77) A.Peled, B.Liu.
"A new hardware realization of high-speed fast fourier Transformers".
IEEE Trans., Vol: ASSP-23, Dec. 1975, p543-547.

(78) B.C.Wang.
"Comments on 'a new hardware realization of digital filters'".
IEEE Trans., Vol: ASSP-25, No: 4, Aug. 1977, p353-354.

(79) Thomas, J.B., B.Liu.
"Error problems in sampling representations".
IEEE Int. Conv. Rec., Pt: 5, 1964, p269-277.

(80) B.Liu, J.B.Thomas.
"Error problems in the reconstruction of signals from sampled data".
Proc. Natl. Electron. Conf., 23, 1967, p803-807

(81) B.Liu.
"Timing jitter in digital filtering".
Proc. 16th Midwest Symp. Circuit Theory, 1, 2.4.1-2.4.10, 1973.

(82) A.V.Balakrishnan.
"On the problem of time jitter in sampling".
IRE Trans., Inf. Theory, April 1962, p226-236.

(83) R.E.Kahn, B.Liu.
"Sampling representations and the optimum reconstruction of signals".
IERE Trans., Inf. Theory, July 1965, p339-347.

(84) A.Papoulis.
"Error analysis in sampling theory".
IEEE Proc., Vol: 54, No: 7, July 1966, p947-955.

(85) J.Katzenelson.
"Errors introduced by combined sampling and quantization".
IRE Trans., Auto. Control, April 1962, p58-68.

(86) W.R.Bennet.
"Spectra of quantized signals".
B.S.T.J. Vol: 27, July 1948, p446-472.

(87) B.Widrow.
"Statistical analysis of amplitude-quantized sampled data system".
AIEE Trans., Appl. Ind., Vol: 79, Jan. 1961, p555-568.

(88) J.B.Knowles, R.Edwards.
"Effects of a finite-word-length computer in a sampled data feedback system".
IEE Proc., Vol: 112, June 1965, p1197-1207.

(89) B.Liu.
"Effect of finite wordlength on the accuracy of digital filters-A review."
IEEE Trans., Vol: CT-18, No: 6, Nov. 1971, p670-677.

(90) A.B.Sripad, D.L.Snyder.
"A necessary and sufficient condition for quantization errors to be uniform and white".
IEEE Trans., Vol: ASSP-25, No: 5, Oct. 1977, p442-448.

(91) A.V.Oppenheim, C.J.Weinstein.
"Effects of finite register length in digital filtering and the Fast Fourier Transform".
IEEE Proc., Vol: 60(8), 1972, p957-976.

(92) D.S.K.Chan, L.R.Rabiner.
"Analysis of quantization errors in the direct form for finite impulse response digital filters".
IEEE Trans., Vol: AU-21, No: 4, Aug. 1973, p354-366.

(93) J.B.Knowles, E.M.Olcayto.
"Coefficient Accuracy and digital filter response".
IEEE Trans., Vol: CT-15, March 1968, p31-41.

(94) E.Avenhaus, H.W.Schuessler.
"On the approximation problem in the design of digital filters with limited word length".
Arch. Elek. Ubertragung. Vol: 24, 1970, p571-572.

(95) E.Avenhaus.
"On the design of digital filters with coefficients of limited word length".
IEEE Trans., Vol: AU-20, No: 3, Aug.1972, p206-212.

(96) Minsoo Suk, Sanjit K. Mitra.
"Computer-aided design of digital filters with finite word length".
IEEE Trans., Vol: AU-20, No: 5, Dec. 1972, p356-363.

(97) A.Hadjifotion, D.G.Appleby.
"Design of digital filters with severely quantized coefficients".
The Radio & Electronic Engineer, Vol: 46, No: 1, Jan. 1976, p23-28.

(98) K.Steiglitz.
"Designing short-word recursive digital filters".
9th Allerton Conf. on Circuit Theory Proc., Oct. 1971, p778-788.

(99) E.Avenhaus.
"An optimization procedure to minimize the word length of digital filter coefficients".
1971 Imperial College Symposium on Digital filtering, London.

(100) W.Schuessler.
"On the approximation problem in the design of digital filter".
Proc., 5th Annual Princeton Conf. On Information Science and Systems, Princeton, N.J., March 1971, p54-63.

(101) R.Hooke, T.A.Jeeves.
"Direct search solution of numerical and statistical problems".
JACM Vol: 8, No: 2, April 1961, p212-229.

(102) B.Gold, C.M.Radar.
"Effects of quantization noise in digital filters".
Proc., AFIPS, Spring Joint Computer Conf., Vol: 28, 1966, p213-219.

(103) S.K.Mitra, R.J.Sherwood.
"Estimation of pole-zero displacements of a digital filter due to coefficient quantization".
IEEE Trans., Vol: CAS-21, No: 1, Jan. 1974, p116-124.

(104) S.R.Parker.
"Limit-cycle oscillations in digital filters".
IEEE Trans., Vol: CT-18, No: 6, Nov. 1971, p687-697.

(105) T.A.Brubaker, J.N.Gowdy.
"Limit cycles in digital filters".
IEEE Trans., Vol: AC-17, No: 5, Oct. 1972, p675-677.

(106) R.B.Kieburtz.
"An experimental study of rounding effects in a 10th-order recursive digital filter".
IEEE Trans., Comm., June 1973, p757-763.

(107) I.W.Sandbery.
"Floating-point roundoff accumulation in digital filter realization".
B.S.T.J. Vol: 46, Oct. 1967, p1775-1791.

(108) C.Y.Kao.
"An analysis of limit cycles due to sign-magnitude truncation in multiplication in recursive digital filters".
Proc. 5th Asilomar Conf., 1971.

(109) T.A.C.M.Claasen, W.F.G.Mecklenbranber, J.B.H.Peek.
"Remarks on the zero-input behaviour of 2nd-order digital filters designed with one magnitude truncation quantizer".
IEEE Trans., Vol: ASSP-23, April 1975, p240-242.

(110) R.B.Blackman.
"Linear data smoothing and prediction in theory and practice".
Addison-Wesley, 1965.

(111) R.B.Kieburtz.
"Rounding and truncation limit cycles in a recursive digital filter".
IEEE Trans., Vol: ASSP-22, Feb. 1974, p73.

(112) I.W.Sandberg, J.F.Kaiser.
"A bound on limit cycles in fixed-point implementations of digital filters".
IEEE Trans., Vol: AU-20, No: 2, June 1972, p110-112.

(113) L.B.Jackson.
"An analysis of limit cycles due to multiplication rounding in recursive digital (sub)filters".
Proc. 7th Allerton Conf. On Circuit And System Theory, 1969, p69-78.

(114) S.R.Parker, S.Hess.
"Canonic realizations of second-order filters due to finite precision arithmetic".
IEEE Trans., Vol: CT-19, July 1972, p410-413.

(115) J.L.Long, T.N.Trick.
"An absolute bound on limit cycles due to roundoff errors in digital filters".
IEEE Trans., Vol: AU-21, No: 1, Feb. 1973, p27-30.

(116) L.B.Jackson.
"Roundoff noise bounds derived from coefficient sensitivities for digital filters".
IEEE Trans., Vol: CAS-23, No: 8, Aug.1976, p481-485.

(117) L.B.Jackson.
"Roundoff noise analysis for fixed-point digital filters realized in cascade or parallel form".
IEEE Trans., Vol: AU-18, No: 2, June 1970, p107-122.

(118) C.T.Mullis, R.A.Roberts.
"Synthesis of minimum roundoff noise fixed-point digital filters".
IEEE Trans., Vol: CAS-23, No: 9, Sept. 1976, p551-562.

(119) W.S.Lee.
"Optimization of digital filters for low roundoff noise".
IEEE Trans., Vol: CAS-21, No: 3, May 1974, p424-431.

(120) L.B.Jackson.
"On the interaction of roundoff noise and dynamic range in digital filters".
B.S.T.J. Vol: 49, No: 2, Feb. 1970, p159-184.

(121) M.Buttner.
"Elimination of limit cycles in digital filters with very low increase in the quantization noise".
IEEE Trans., Vol: CAS-24, No: 6, June 1977, p300-304.

(122) A.C.Callahan.
"Random rounding, some principles and applications".
Proc. 1976 IEEE Int. Symp. Acoust, Speech and Signal Processing p501-504.

(123) M.Buttner.
"A novel approach to eliminate limit cycles in digital filters with a minimum increase in the quantization noise".
Proc. 1976 IEEE Int. Symp. Circuits and Systems, p291-294.

(124) T.Claasen, W.F.G.Mecklenbranker, J.B.H.peek.
"Frequency domain criteria for the absence of zero-input limit cycles in non-linear discrete-time systems with applications to digital filters".
IEEE Trans., Vol: CAS-22, No: 3, March 1975, p232-239.

(125) A.I.Barkin.
"Sufficient conditions for the absence of Auto-oscillations in pulse systems".
Automat. Remote Contr., Vol: 31, June 1970, p942-946.

(126) E.D.Garber.
"Frequency criteria for the absence of periodic responses".
Automat. Remote Contr., Nov. 1967, Vol: 28.

(127) P.M.Ebert et.al.
"Overflow oscillations in digital filters".
B.S.T.J. Vol: 48, Nov. 1969, p2999-3020.

(128) A.N.Willson, Jr.
"Some effects of quantization and adder overflow on the forced response of digital filters".
B.S.T.J. Vol: 51, No: 4 April 1972, p863-887.

(129) A.N.Willson, Jr.
"Limit cycles due to adder overflow in digital filters".
IEEE Trans., Vol: CT-19, No: 4, July 1972, p342-346.

(130) T.A.C.M.Classen, L.O.G.Kristiansson.
"Necessary and sufficient conditions for the absence of overflow phenomenon in a second-order digital filter".
IEEE Trans., Vol: ASSP-23, No: 6, Dec. 1975, p509-575.

(131) I.W.Sandberg.
"A theorem concerning limit cycles in digital filters".
Proc. 7th Allerton Conf. On Circuit And System Theory, 1969, p63-67.

(132) B.Gold, T.Bially.
"Parallelism in Fast Fourier Transform Hardware".
IEEE Trans., Vol: AU-21, Feb. 1973, p5-16.

(133) H.L.Groginsky, G.A.Works.
"A pipeline Fast Fourier Transform".
IEEE Trans., Computer Vol: C-19, Nov. 1970, p1015-1019.